Diversity Curves for Graph Representation Learning

Katharina Limbeck[1,2] **Nadja Häusermann**[3] **Martin Carrasco**[3]
Guy Wolf[†,4,5] **Bastian Rieck**[†,3]

[1]Helmholtz Munich [2]Technical University of Munich [3]University of Fribourg
[4]University of Montréal [5]Mila - Quebec AI Institute

Abstract

Graph-level representations are crucial tools for characterising structural differences between graphs. However, comparing graphs with different cardinalities, even when sampled from the same underlying distribution, remains challenging. Unsupervised tasks in particular require interpretable, scalable, and reliable size–aware graph representations. Our work addresses these issues by tracking the structural diversity of a graph across coarsening levels. The resulting graph embeddings, which we denote *diversity curves*, are interpretable by construction, efficient, and directly comparable across coarsening hierarchies. Specifically, we track the *spread* of graphs, a novel isometry invariant that is inherently well-suited for encoding the metric diversity and geometry of graphs. We utilise edge contraction coarsening and prove that this improves expressivity, thus leading to more powerful graph-level representations than structural descriptors alone. Demonstrating their utility over a range of baseline methods in practice, we use *diversity curves* to (i) cluster and visualise simulated graphs across varying sizes, (ii) distinguish the geometry of single-cell graphs, (iii) compare the structure of molecular graph datasets, and (iv) characterise geometric shapes.

1 Introduction

Understanding structural differences between graphs is a key step in exploring, describing, and distinguishing graph datasets or distributions. In particular when comparing graphs drawn from the same distribution, suitable graph representations should adapt to differences in scale and capture coarse structural similarities between graphs without relying on supervision. However, many existing methods for graph comparison and graph-level representation learning are inherently sensitive to differences in graph sizes [82, 96]. This can negatively impact interpretation and downstream performance. We address this issue by investigating to what extent *graph coarsening* techniques can be used to perform structural comparisons between graphs. To this end, we propose *size-aware methods* that are aware of both the cardinality of graphs, as well as their intrinsic size, i.e. their structural diversity across coarsening scales. These size-aware representations aim to address geometric tasks, where graphs have been sampled from the same underlying distribution and share the same inherent structure but vary in size. Our work makes the following **contributions**:

Novel graph representations: We propose *diversity curves* for comparing graphs. Our approach is the first to use coarsening together with structural notions of diversity to derive hierarchical graph descriptors that track how graph structure evolves across the coarsening hierarchy.

Expressivity guarantees: We prove that edge contraction coarsening improves the expressivity of graph structural comparisons via graph invariants.

[†]These authors jointly supervised this work.

40th Conference on Neural Information Processing Systems (NeurIPS 2026).

Excellent performance: We show that diversity curves excel at geometric tasks, outperforming unsupervised baseline methods across a range of applications. Our methods perform well at clustering simulated graph distributions, distinguishing between trajectory- and cluster-like single-cell graphs, and characterising geometric graphs across varying sizes.

2 Related work

The problem of finding *size-robust graph representations* has been studied from different viewpoints. Recent research studying the size-robustness has shown that graph neural networks (GNNs), typically used in supervised settings, struggle with size generalisation in certain instances, for example when learning solutions to combinatorial problems [38], or characterising geometric graphs across varying scales [3]. Generalisation across coarsening resolutions and graph sizes has further been shown to be difficult for GNNs applied to molecular graphs [41], or across specific graph distributions [96]. Learning size-robust graph representations thus is a core topic of interest for graph learning.

Unsupervised methods. Size generalisation remains relevant for unsupervised methods, which are the focus of our work. As commonly used baselines, graph kernel methods based on path- or walk-based kernels are known to suffer from size-sensitivity [29], making their embeddings less suitable in unsupervised settings. Similar size-sensitivity issues have been observed in *unsupervised graph embeddings*. For instance, graph2vec compares subgraph collections [62], which is more nuanced than directly comparing entire graphs, but can still be dominated by the vocabulary of larger graphs. Spectral embeddings, such as NetLSD [90] or SpectralZoo [36], aim to characterise each graph via summarising the spectrum of eigenvalues of its normalised Laplacian. In practice, such methods still struggle with clustering graphs generated from the *same* model across varying densities and sizes [82]. Other approaches like FEATHER [74] leverage random walks, which are pooled to obtain graph-level embeddings, thus also implicitly incorporating size. Moreover, while contrastive learning can be used to adapt GNNs to unsupervised tasks [97, 98], their reliance on graph augmentations and comparisons between positive and negative pairs limits their training efficiency and their training objective is sensitive to varying graph sizes.

Graph filtrations. To obtain interpretable representations, prior work on geometric and topological machine learning often employs the concept of *graph filtrations*. Such filtrations utilise nested sequences of subgraphs derived from different graph descriptors to obtain a multiscale view of a graph. Across these filtrations, persistent homology (PH) tracks topological features [2, 81], while TopER reports the node and edge counts [86], and filtration curves evaluate general graph descriptors [63]. Many of these approaches remain inherently size-sensitive because the subgraph filtrations and final quantities recorded often explicitly or implicitly depend on the size of the input graph [10, 80]. This motivates the need to develop size-robust geometric representations, potentially breaking with the ongoing paradigm of relying on subgraph filtrations, and considering novel construction processes.

Magnitude and spread. To find size-robust representations, we propose a novel approach leveraging *magnitude* and *spread* as graph invariants that measure the structural diversity of graphs, i.e. the effective number of distinct vertices, based on node distances [47, 93]. These measures were originally introduced to assess the diversity of a metric space across different scales of the metric; recently, they have seen use in deep learning applications, specifically for evaluating the diversity of latent representations across varying distance scales [53], or for obtaining geometry-aware edge pooling schemes [54]. Our work uses the *spread*, as it is computationally more efficient than magnitude, while being highly correlated [54, 93]. Spread in particular has not yet been studied in the context of comparing graphs of various cardinalities, enabling us to advance beyond the previously proposed distance-scaled summaries of diversity towards developing novel size-aware geometric tools.

Graph coarsening. Our method leverages the use of a *coarsening method* to track the diversity of graphs at different cardinalities. Graph coarsening is commonly used in hierarchical pooling layers of GNNs that aim to reduce the size of graphs while preserving graph structure [25, 55]. We exploit this approach for our diversity curves, using *edge contraction coarsening* specifically because it is an iterative pooling operation that preserves graph structure [54] while improving model expressivity [46]. Combining diversity invariants together with edge pooling in particular offers promising potential for developing size-robust and expressive graph-level representations.

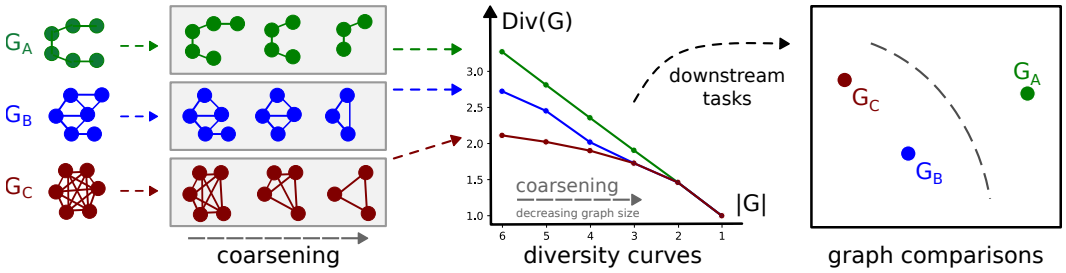

Figure 1: Diversity curves for three example graphs. Our framework first *coarsens* the initial graphs via iterative edge contractions and tracks structural invariants across the coarsening process. Specifically, we compute structural diversity based on a choice of metric on the graph, e.g. shortest-path distances. *Diversity curves* then represent the structural diversity of a graph as a function of the number of nodes in the coarsened graph. The resulting vector representations compare the coarsened geometry of graphs for unsupervised visualisation, clustering, classification, or explanatory tasks.

3 Methods

We propose a framework to compare the structural diversity of graphs across a coarsening process. The resulting diversity curves yield a vectorised representation for each graph, encoding their hierarchical geometry, which can be used for downstream analysis. Figure 1 illustrates the construction of diversity curves, Section 3.1 describes the process and Section 3.2 their properties.

3.1 Diversity curves as graph representations

Graph coarsening as a multiscale process. To compare graphs across cardinalities, we coarsen or upsample graphs to the same range of node numbers. This requires a graph coarsening process that (i) allows us to control the number of nodes in the coarsened graphs, (ii) is iterative, (iii) is applicable to unattributed graphs, (iv) is unsupervised, and (v) preserves the connected components in the graph. Subsequently, we use edge contraction coarsening, which satisfies our requirements, in particular permits controlling the size of coarsened graphs, is known to improve the expressivity of GNNs [46], is an interpretable operation, and does well at preserving the structure of graphs [54]; refer to Section B.1.1 for an extended discussion on the coarsening operation. Formally, we select a set of cardinalities, $\mathcal{I} \subset \mathbb{N}$, across which we want to compare the coarsened geometry of graphs and we consider a graph $G = (X, E)$ to consist of a set of vertices X and a set of edges E. We use the notation $|G| = |X| = n$ for the cardinality of the graph and $m = |E|$ for the number of edges. Furthermore, we define a sequence $(e_i)_{1 \leq i \leq k}$ of edges in G that are to be contracted, for some $k \in \{1, \dots, m\}$ such that $n - k = \min_{i \in \mathcal{I}} i$. The edge sequence induces a sequence of coarsened graphs $(G_i)_{n-k \leq i \leq n}$ based on contracting these edges in order, so that $G_n = G$ and $G_{i-1} = G_i / e_{n-i+1}$. Note that $|G_i| = i$. To apply the coarsening operation in practice, we follow existing implementations of edge contraction methods [54] to select the sequence of edges to contract. Specifically, we obtain this edge sequence by assigning a score to each edge and iteratively contracting the edges starting with the lowest scores, while contracting no edge adjacent to a previously contracted edge. If needed, we reapply this procedure to coarsen graphs to the desired size and aggregate the features of any merged nodes, for instance by averaging. We consider scoring based on each edge's importance for the graph's structure [54] or random scoring (cf. Section B.1.1). In case a dataset contains small graphs, we further employ upsampling strategies, for example by uniformly upsampling a node and reconnecting it to all neighbours of the initial node. This strategy allows us to extend the sequence of coarsened graphs to $(G_i)_{n-k \leq i \leq N}$ for some $N > n$.

Structural diversity for comparing graphs. Structural diversity is inherently well-suited to compare graphs at the same cardinality. By measuring diversity as the *effective number* of distinct nodes in a graph, the *spread* of a graph is a novel and expressive invariant that encodes the graph's geometry based on a choice of distance between nodes. Throughout this work, we consider a graph $G = (X, E)$ to be associated with a metric d, so that (X, d) defines a finite metric space. Given such

a graph, its structural diversity, as measured via the *spread* of its metric space [54, 93], is defined as

$$\mathrm{Div}(G) := \sum_{x \in X} \frac{1}{\sum_{y \in X} \exp(-d(x,y))}. \tag{1}$$

Note that $\mathrm{Div}(G) \in [1, |X|]$ and that structural diversity can be directly interpreted as the effective number of nodes in the graph. Through coarsening, we aim to exploit the inherent link between diversity and this notion of effective size. Spread is more efficient to compute than alternative diversity measures like the magnitude of a metric space [47, 93], while allowing for a flexible choice of geometry determined by the metric on the graph. Further, spread can be directly compared across graphs as it is invariant to isometries and permutations between nodes [54]. By default, we use shortest-path distances for diversity computations as we find that they perform well across our experiments and are faster to estimate than alternative graph metrics, say diffusion distances or resistance distances, as explored in the ablation study in Section D.8.1. Coarsening allows structural invariants to be robust to varying cardinalities. In fact, coarsening counteracts one of the notable weaknesses of many geometric tools, diversity measures, or path-based summaries, which are known to struggle with directly comparing data of varying sizes [53]. Our framework thus still allows us to use shortest-path distances to make comparisons across coarsened graphs. Extending our framework, other choices of graph invariants or summary statistics are possible, but we consider spread to be a useful and flexible default choice for the reasons described above.

Diversity curves as size-aware graph representations. Our final graph representations, which we call *diversity curves*, track the structural diversity of each graph as a function of the number of nodes in the coarsened graph. Let G be a graph associated with a metric d. Further, fix a sequence of integers $\mathcal{I}$ representing cardinalities, and a choice of down- or upsampling strategy. Using the down- or upsampling methods we construct a multiset of sequences of coarsened graphs, $\mathcal{G} = \{\{(G_{i,j})_{i \in \mathcal{I}}\}\}_{j \in \mathcal{J}}$ indexed by some set $\mathcal{J}$ such that $|G_{i,j}| = i$ and $G_{n,j} = G$ for all $j \in \mathcal{J}, i \in \mathcal{I}$. Then, we define the corresponding *diversity curve*, $\mathrm{DivCurve}(G) = \mathrm{DivCurve}(\mathcal{G}) \in \mathbb{R}^{|\mathcal{I}|}$, at cardinality $i \in \mathcal{I}$ by

$$\mathrm{DivCurve}(G)_i := \frac{1}{|\mathcal{J}|} \sum_{j \in \mathcal{J}} \mathrm{Div}(G_{i,j}), \tag{2}$$

the mean diversity of all coarsened graphs in $\mathcal{G}$ of cardinality i. This yields a vector representation for each graph, which describes the graph's coarsened geometry across resolutions. Intuitively, graphs with few localised differences will behave similarly when contracted and have more similar diversity curves than graphs with globally different structures. Diversity curves capture exactly these hierarchical trends. In fact, the value of the diversity curve at each scale directly summarises how structurally diverse a given graph is, leading to a clear interpretation as the effective number of distinct nodes in the coarsened graph. This gives a multiscale and size-aware summary of graph structure and diversity that is robust to differences in graph size. Our methods thus improve upon previous work on geometric machine learning by proposing the combination of coarsening and diversity as a size-aware alternative to existing graph filtrations. Diversity curves can be directly treated as functions or vectorised summaries, which can be stored efficiently, and allow for flexible comparisons between graphs. This offers an alternative to graph-level embeddings. Further, we can compute pairwise distances between curves via the L^p-norm between them, allowing us to make formal distributional comparisons between graphs.

Default parameter choices. We set the coarsening scales $\mathcal{I}$ at which we compute diversity curves to all integers between 1 and the maximum number of nodes in the dataset, but other choices are possible. For example, choosing fewer and lower evaluation scales can ensure efficient computations (cf. Section D.8.3). We compute the diversity curves by default over a set of three coarsening sequences using random edge contractions as we find the choice of edge scoring has no notable impact on performance (cf. Section D.8.1) and the diversity curve is reasonably stable with respect to the randomness of the edge score and the number of repeats (cf. Section D.8.2). This is motivated by the fact that while we do not immediately know which coarsening sequence best describes a given graph, we aim to capture key structural characteristics through repeated coarsening. In fact, we will prove how edge pooling improves expressivity. By default we compute diversity using shortest-path distances and set $\mathrm{DivCurve}(G)_1 = 1$ as a graph with one node effectively has one distinct point [73, 93]. Further, we cannot use edge contraction pooling to reduce the size of a graph to less than the number of connected components. In these cases, we use linear interpolation of the

diversity curves to estimate the value of diversity at lower sizes. We publish code for our methods on GitHub and detail our algorithm in Section B.

3.2 Properties of diversity curves

In the following, we describe the theoretical properties that ensure diversity curves are well-behaved and expressive graph representations. Complete proofs and further results are provided in Section C. For some of the theoretical treatment, we consider the diversity curve of a graph over the multiset of all possible edge contraction sequences. This allows us to talk about the invariance and expressivity of diversity curves from a theoretical perspective without the need for a canonical labelling of the edges, a notoriously difficult problem related to the graph isomorphism problem [1].

Diversity curves as graph invariants. Our measure of structural diversity is invariant under permutations of vertices and under isometries [93]. For two isomorphic graphs $G = (X, E)$ and $H = (Y, F)$, with edges $e \in E$ and $f \in F$ that are identified under the isomorphism, the graphs G/e and H/f are also isomorphic and diversity curves are invariant in the following sense.

Theorem 1. *For graphs $G \cong H$, their diversity curves over all possible edge contractions are equal.*

Diversity curves collapse to the same diversity. We investigate the limiting behaviour of diversity curves at low coarsening scales by considering the properties of structural diversity and the requirement posed on the coarsening method to preserve connectivity. The result below shows that when coarsening graphs up to cardinality 1, they eventually all have the same diversity. As such, diversity curves can distinguish the number of connected components of a graph and behave reasonably under increased coarsening.

Theorem 2. *Any two graphs G and H with the same number c of connected components will collapse to the same graph of cardinality c and their diversity curves agree for $i \in \{1, \dots c\}$. Concretely,* $\mathrm{DivCurve}(G)_i = \mathrm{DivCurve}(H)_i = i$. *Furthermore,* $\mathrm{DivCurve}(H)_{c+1} = \mathrm{DivCurve}(G)_{c+1} \neq c + 1$.

Diversity curves increase expressivity. Expressivity, that is the ability to distinguish between graphs, is a key property of useful graph representations [92]. To motivate the use of diversity as a graph invariant together with the coarsening process, we show that expressivity is improved when combining them, following the definition of increasing expressivity by Lachi et al. [46], restricting it to consider graphs of the same cardinality. Hence, we show that diversity curves improve upon the expressivity of diversity as measured by the spread of a graph through edge contraction pooling.

Definition 1. *For two functions φ and ψ on the set of graphs of a fixed cardinality, we say that φ is* more expressive *than ψ if for any graphs G_A, G_B such that $\psi(G_A) \neq \psi(G_B)$, then also $\varphi(G_A) \neq \varphi(G_B)$, and there exist graphs G_C, G_D such that $\varphi(G_C) \neq \varphi(G_D)$ and $\psi(G_C) = \psi(G_D)$.*

Theorem 3. *The diversity curves over all possible edge contractions is more expressive than spread.*

The proof of Theorem 3 consists of providing a pair of graphs with identical spread but distinguishable by their diversity curves. In Section 4.6 we further demonstrate empirically that this holds not just for one example, but improves expressivity across a wide selection of graphs.

Computational complexity. The time complexity of computing the diversity curve of a graph $G = (X, E)$ is determined by $C_{\text{Coarse}} = \mathcal{O}(|E|(\log(|E|) + |X|) + n_{\text{max}}^2)$, constructing one sequence of coarsened graphs using random edge contractions and upsampling, and $C_{\text{Div}} = O(|\mathcal{I}| \cdot n_{\text{max}}^3)$, evaluating the structural diversity of all coarsened graphs, where $n_{max} = \max_{i \in \mathcal{I}} i$. Altogether, the time complexity of computing the diversity curve is given by $C_{\text{Coarse}} + C_{\text{Div}}$. Section C.1 contains a more detailed description of the theoretical time complexity, while Section D.7 evaluates the empirical costs; we conclude that our methods are efficient to compute while outperforming more costly methods, such as the graph filtrations used by TopER [86], on large graphs. Because coarsening is an integral component of our framework, computational costs can be efficiently reduced by preprocessing graphs to smaller sizes, or evaluating diversity curves at fewer evaluation scales or lower cardinalities. Furthermore, the computation of the diversity at the different scales can be parallelised. We discuss different options for improving the efficiency of our methods in Section C.1.

4 Experiments

We show the following experimental results: Section 4.1 verifies that the changes in diversity curves induced by edge perturbations correlate with the strength of the perturbation. This confirms that our methods capture differences between modified graphs. Next, Section 4.2 demonstrates that diversity curves perform well at distinguishing graph distributions and parameter choices while being robust to variations in graph sizes. Based on this, we investigate applications to bioinformatics tasks. Section 4.3 shows that diversity curves distinguish between the coarse geometry of trajectory- and cluster-like single-cell graphs surpassing more complex summaries. Section 4.4 uses diversity curves to characterise similarities between enzyme graphs through statistical comparisons. Section 4.5 further confirms that diversity curves reach high performance at distinguishing geometric shapes. Section 4.6 reports empirical expressivity results supporting our theoretical analysis. Overall, this demonstrates that diversity curves constitute useful and expressive methods for unsupervised tasks.

4.1 Diversity curves track graph perturbations

As a first correctness check, we investigate how diversity curves behave under perturbations of the input graphs [66]. We fix a degree of perturbation $p \in [0, 1]$ and apply edge perturbations by consecutively adding, removing, swapping, or rewiring edges, resulting in edits to all graphs in the SBM, PLANAR, EGO, LOBSTER and COMMUNITIES datasets [44, 66]. For each dataset and degree of perturbation, we compute the mean norm across diversity curves, which summarises the structural diversity across graphs. We expect that the more graphs are perturbed, the larger the change in diversity curves should be. As demonstrated in Figure 2, this trend is true in practice: Across almost all datasets and perturbation types, the more edges get perturbed, the larger the change in diversity curves as measured by the Spearman correlation between the norm of the curves and the degree of perturbation. When adding edges, nodes become more similar and structural diversity decreases, hence we observe almost perfect *negative correlation*. When removing edges, distances between nodes and the value of diversity curves increases, leading to perfect *positive correlation*. Rewiring and swapping edges similarly tends to destroy graph structure and increase structural diversity with high correlations for almost all datasets other than PLANAR, which by construction contains more random connectivities. Additional results in Section D.1 confirm that diversity curves detect small perturbations that result in structural changes to graphs.



Figure 2: Spearman correlation between the degree of perturbation and the change in diversity curve per dataset.

4.2 Diversity curves distinguish graph distributions

To investigate size-robustness, we test whether our method can distinguish graphs from different distributions, namely Erdős-Rényi (ER), stochastic block model (SBM), random partition (RP), and random geometric (RG) graphs. For each model and graph sizes between 10–29 nodes, we generate three graphs using a fixed parameter setting (cf. Section D.2). We compute diversity curves using shortest-path distance, using the L^2-norm of the representations to visualise distributions with PCA. Figure 3 shows that our embedding clearly separates graphs from different distributions. Further, we use the diversity curves for 10-fold stratified grouped CV using a kNN-classifier to distinguish models. We compare to relevant unsupervised graph representation methods, i.e. graph kernels [79], spectral graph representations via SpectralZoo [36] and NetLSD [90], topology-inspired embeddings via TopER [86] and vectorial representations of persistence images computed via persistent homology [2], as well as a random walk based approach, FEATHER [74]. Results in Table 1 demonstrate that diversity curves perform similarly to the best baselines in terms of classification accuracies. To further investigate the quality of each graph representation, Table 1 compares *silhouette scores*, obtained from 50% of the data over 100 resamples, as a measure of clustering quality, for which diversity curves *outperform* all other methods, confirming that diversity curves achieve better clustering than baseline methods and demonstrating superior performance at distinguishing graph distributions in an unsupervised manner while being robust to differences in graph sizes.

Table 1: Performance (mean ± std) of unsupervised methods at distinguishing graphs from four different distributions.

Method	Accuracy (↑)	Silhouette Score (↑)
Diversity Curves (ours)	**0.90 ± 0.06**	**0.42 ± 0.03**
Kernel WL Opt. Assig.	**0.90 ± 0.05**	0.02 ± 0.00
Kernel Shortest Path	0.77 ± 0.07	0.28 ± 0.04
TopER	0.70 ± 0.10	–0.06 ± 0.02
NetLSD	**0.90 ± 0.03**	0.06 ± 0.04
FEATHER	0.75 ± 0.09	0.11 ± 0.02
SpectralZoo	0.72 ± 0.09	0.05 ± 0.03
PH Vietoris–Rips	0.78 ± 0.07	–0.01 ± 0.03

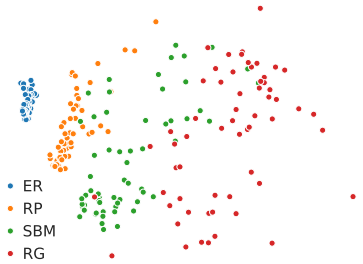

Figure 3: PCA of diversity curves for ER, RP, SBM, and RG models.

Next, we test the ability of our method to characterise and distinguish between graphs generated from the *same* underlying random graph model using three different parameter choices. For each graph distribution, parameter setting (cf. Appendix D.2), and graph sizes from 10–29 nodes, we generate three graphs. We again use the Euclidean distances of the diversity curves with the shortest-path distance as input to a kNN-classifier with 10-fold stratified grouped CV. The results reported in Table 2 demonstrate that diversity curves achieve high classification accuracies. We compare silhouette scores in Figure 4 to our baselines, and achieve silhouette scores comparable to the best kernel baselines for the ER and RP distributions while obtaining the highest scores for RG and SBM graphs. This confirms that diversity curves outperform other unsupervised methods at distinguishing between parameters used in a random graph model, while being robust to size differences. Additional clustering quality metrics and extended results in Appendix D.2 confirm our findings.

Table 2: Accuracies (mean ± std) at distinguishing parameter choices of random graph models. Diversity curves reach top performance scores across graph distributions.

Method	ER	RP	RG	SBM
Diversity Curves (ours)	**1.00 ± 0.00**	**0.98 ± 0.04**	**0.97 ± 0.04**	**0.86 ± 0.08**
Kernel Shortest Path	**1.00 ± 0.00**	0.97 ± 0.04	0.89 ± 0.13	0.64 ± 0.08
Kernel WL Opt. Assig.	0.98 ± 0.04	0.89 ± 0.08	0.93 ± 0.06	0.80 ± 0.08
TopER	0.89 ± 0.08	0.87 ± 0.09	0.80 ± 0.10	0.60 ± 0.07
NetLSD	0.97 ± 0.05	0.97 ± 0.04	0.92 ± 0.07	0.83 ± 0.08
FEATHER	**1.00 ± 0.00**	0.86 ± 0.08	0.94 ± 0.07	0.66 ± 0.12
SpectralZoo	**1.00 ± 0.00**	0.91 ± 0.07	0.91 ± 0.11	0.63 ± 0.08
PH Vietoris–Rips	**1.00 ± 0.00**	0.89 ± 0.08	0.94 ± 0.08	0.61 ± 0.10

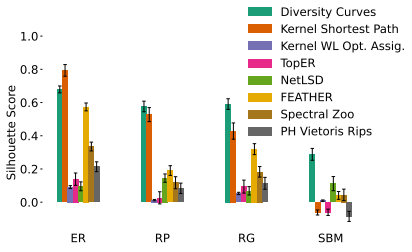

Figure 4: Silhouette scores (↑) for distinguishing different parameter choices.

4.3 Diversity curves distinguish trajectory or cluster structures in single-cell graphs

Comparing graphs of vastly different sizes is crucial when analysing single-cell transcriptomics data, where the number of cells often varies widely due to experimental factors. A key question to inform further downstream analysis is whether single-cell graphs show cluster- or trajectory-like structures. This inherently geometric question informs the biological interpretation of data as well as the choice of computational tools and visualisation methods designed for either type. To address this task, we consider a collection of 169 curated single-cell graphs that have been annotated as cluster- or trajectory-like using biological ground truths [52, 76]. For each graph processed as described in Section D.3, we compute the diversity curve using shortest-path distances and the diversity curve using Euclidean distances between node features, concatenate both curves, and use these curves as our graph representations. We note that cluster-like graphs have on average higher structural diversity than trajectory-like graphs. The PCA plot in Figure 5 further shows that our graph representation separates well between both types of graph structure. We compare to the methods of Lim and Qiu [52], who compute a set of five geometric scores: the entropy of pairwise diffusion pseudo-time geodesic distances, an entropy-based persistent homology score, a summary of vector norms, a summary of Ripley's K function, and a summary of reachability values. To our knowledge, these are the only methods that have so far been investigated for this task. We use the geometric scores as inputs for 5-fold stratified CV using a kNN-classifier. Table 3 reports classification accuracies compared to the same classification results when using our diversity curves, mean pooled node features, or alternative graph embeddings as inputs. We achieve notably better accuracies using diversity curves

than the collection of scores by Lim and Qiu [52]. Hence, we find that diversity curves provide more useful and more interpretable graph representations than existing methods designed for this task. Further, we confirm that our methods perform well compared to other unsupervised graph embedding methods. We thus show the potential of diversity curves for distinguishing the geometry of single-cell graphs, giving actionable insights into deciding further analysis steps.

Table 3: Performance at distinguishing between trajectory- and cluster-like single-cell graphs.

Dataset	Method	Test Accuracy (↑)
Gold (87 Graphs)	Diversity Curves (ours)	**0.860 ± 0.077**
	NetLSD	0.823 ± 0.047
	Scores ([52])	0.804 ± 0.053
	Best Kernel	0.785 ± 0.079
	TopER	0.672 ± 0.086
	Node Features	0.661 ± 0.091
All (169 Graphs)	Diversity Curves (ours)	**0.759 ± 0.077**
	NetLSD	0.731 ± 0.047
	Scores ([52])	0.716 ± 0.053
	Best Kernel	0.685 ± 0.092
	TopER	0.624 ± 0.086
	Node Features	0.619 ± 0.091
	Landscape ([52])	0.570 ± 0.025

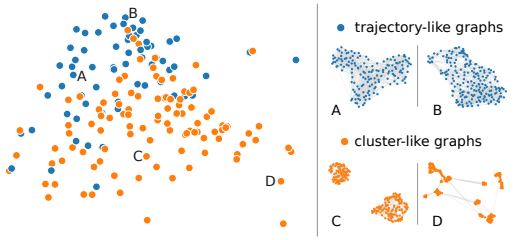

Figure 5: PCA visualisation of the diversity curves for all 169 single-cell graphs (left). Examples of two trajectory-like graphs and two cluster-like single-cell graphs (right).

4.4 Diversity curves characterise molecular graph datasets

Characterising the coarse geometry of graphs, diversity curves can be used to compare and find structural similarities between graph datasets. Specifically, we explore common trends across classes of molecular graphs from the ENZYMES or PROTEINS datasets [6], where each graph represents a protein, with some proteins also being enzymes. We first compare the mean diversity curves per class, observing that non-enzyme graphs are notably distinct and have lower structural diversity on average than enzyme graphs. We further implement a non-parametric two-sample permutation test for the difference in mean diversity curves using the L^2-norm between the curves as a test statistic. For each comparison across classes, Figure 6 reports the p-values, which we computed across 1000 random permutations of the data and by adding a pseudo-count for error control [69]. The results confirm that the set of non-enzymes is clearly distinguishable from all enzyme classes. Meanwhile, we find no evidence to indicate that enzymes from the PROTEINS dataset differ from the EC4 or EC1 classes in the ENZYMES dataset through their averaged diversity curves. This verifies that the geometry of these proteins is more coarsely similar to enzymes than to non-enzymes, even when comparing across datasets. Hence, diversity curves show that enzymes that share similar biological functions have the same structure across different coarsening scales and datasets, which is a crucial question for evaluating graph learning datasets [14], and a step towards quantifying structural similarities for transfer learning tasks [34].

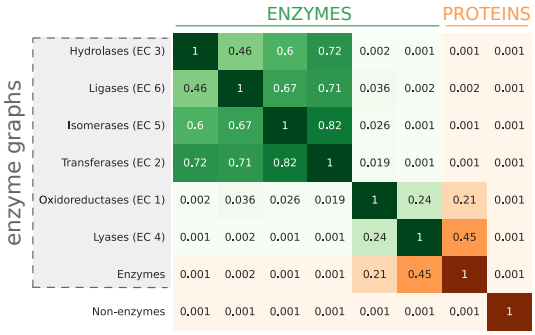

Figure 6: Two sample permutation testing results (p-values) for testing the equality in mean diversity curves per class in the PROTEINS and ENZYMES datasets using the L^2-norm between curves.

4.5 Diversity curves characterise geometric shapes

Having investigated bioinformatics applications, we further perform an experiment to support our claim that diversity curves are capable of describing purely geometric graphs. To this end, we consider the MANTRA dataset [3], which contains, among others, triangulations of 2-manifolds, i.e. Klein bottle, torus, sphere, or the real projective plane. The task of distinguishing these manifolds has been shown to be challenging for graph-based models in particular [3]. Following our established pipeline, we compute diversity curves using shortest-path distances on the 1-skeleton graphs of the triangulation and use these representations as inputs for SVM classification. These graphs range from 3 to 15 nodes. We denote methods that take graphs or triangulations as inputs by G and $\mathcal{T}$, respectively, in

Table 4. Because Klein bottles and tori have the same 1-skeletons, an optimal graph classifier could reach up to 73.8% accuracy. Our method (cf. Table 4) almost reaches this theoretical upper bound. Next, we extend our approach and compute diversity curves from heat kernel distances between faces computed from the Hodge Laplacian [43], which encodes higher-order structures. Following the same classification procedure as before, the combined diversity curves almost completely solve the task and reach 99% accuracy at distinguishing the homeomorphism types. This is superior to the best baseline reported by Ballester et al. [3], i.e. the *cellular transformer* (CT), indicating our representations can be used for mesh data and shape tasks. We further show a PCA plot of the combined diversity curves in Figure 7, which indicates that our methods effectively separate classes. Finally, we investigate whether our representations encode intrinsic geometric information that is robust to varying graph sizes. To do so, we train our classifier on the diversity curves of the original dataset, but test on a modified dataset created by barycentric subdivision of the original triangulations [3], denoted by $\mathcal{T}_{\text{up}}$, giving graphs of cardinalities between 14 and 165. As reported in Table 4, our methods *retain* high classification performance surpassing alternative models at this arguably difficult experimental setup. We conclude that diversity curves are expressive representations that effectively characterise geometric graphs across varying resolutions and sizes.

Table 4: Model performance (mean ± std) at distinguishing homeomorphism types of triangulations of 2-manifolds from MANTRA.

Method	Version	Accuracy (↑)	AUROC (↑)
Diversity Curve (ours)	$\mathcal{T}$	**0.99 ± 0.00**	**1.00 ± 0.00**
CT (best result from [3])	$\mathcal{T}$	0.91 ± 0.01	0.94 ± 0.00
Diversity Curve (ours)	G	**0.74 ± 0.00**	**0.85 ± 0.00**
Optimal Classifier	G	**0.74 ± 0.00**	0.80 ± 0.00
Diversity Curve (ours)	$\mathcal{T}_{\text{up}}$	**0.83 ± 0.02**	**0.96 ± 0.01**
CT (best result from [3])	$\mathcal{T}_{\text{up}}$	0.74 ± 0.00	0.83 ± 0.01

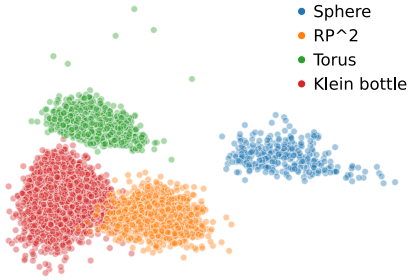

Figure 7: PCA visualisation of diversity curves for MANTRA.

4.6 Diversity curves are expressive graph representations

To support our theoretical expressivity analysis, we investigate whether our methods can distinguish pairs of WL-indistinguishable graphs from the BREC datasets [46, 92]. The results in Table 5 confirm that diversity curves improve the expressivity of structural diversity measures through graph coarsening. On its own, structural diversity based on shortest-path distances can distinguish 19% of graphs in BREC through their absolute difference in spread within an error tolerance of 10^{-5}. Further, by comparing diversity curves evaluated for all possible ways of contracting one edge, we increase the proportion of graph pairs that can be distinguished to 50.8%. This demonstrates the practical utility of using edge pooling for increasing expressivity. Further, these insights highlight that diversity curves successfully distinguish 1-WL indistinguishable graphs and are comparable in expressivity on the BREC dataset to NetLSD while being clearly more expressive than persistent homology with Vietoris-Rips filtrations and FEATHER. Comparisons with further baselines and detailed results for the different classes of graphs in BREC can be found in Section D.6.

Table 5: Number of pairs from the whole BREC dataset distinguished via structural diversity or diversity curves. Compared to representative results from [2, 81, 92].

Method	All (400)
Diversity Curves	203 (50.8%)
Div(G)	76 (19.0%)
3-WL [92]	270 (67.5%)
1-WL [81]	0 (0.0%)
Graphormer [92]	79 (19.8%)
EdgePool [46]	23 (5.8%)
PH Vietoris-Rips from [2]	84 (21.0%)
NetLSD [75, 90]	213 (53.2%)
FEATHER [74, 75]	103 (25.8%)

4.7 Extended results and ablation studies

In addition to our main results, we empirically evaluate the computational efficiency of our methods on large graphs in Section D.7, noting that through coarsening, diversity curves remain scalable to graphs up to 2^{12} nodes. Section D.9 shows classification results on MCF-7 and MOLT-4, two large molecular graph datasets from TU datasets [59], across which diversity curves show competitive performance compared to the unsupervised baseline methods evaluated throughout our work.

Further extending the results from Table 1, we observe that diversity curves clearly outperform simple structural baselines at distinguishing graph distributions as detailed in Section D.2, verifying that diversity curves capture information beyond basic graph statistics. To clarify the impact of different parameter choices on the performance of our methods, we provide further ablations, described in more detail in Section D.8. First, we investigate the impact of the selected coarsening strategy and graph metric on the experiment for distinguishing parameter choices as reported in Figure 4. The results in Section D.8.1 show that diversity curves computed using shortest-path distances and random edge contraction coarsening reach high silhouette scores compared to alternative metric choices, justifying the utility of our default choice. Next, we investigate the influence of varying the number of evaluation scales on the experiment on distinguishing graph distributions from Table 1, noting that when we decrease the number of scales runtime decreases and accuracy remains stable, only decreasing for lower numbers of scales as reported in Table 6. We further investigate the interplay between the up- and downsampling strategies by considering the operations individually in Section D.8.4 and find that the strength of our methods lies in being able to combine both up- and downsampling strategies to compare graphs in a dataset at the same cardinalities. Finally, in Section D.8.2, we vary the number of edge contraction sequences over which we evaluate diversity curves for this experiment, noting that task performance and the diversity curves themselves are reasonably stable across different edge sequences.

Table 6: Effect of the number of evaluation scales ($|\mathcal{I}|$) on runtime and accuracy.

$\lvert\mathcal{I}\rvert$	Runtime (s)	Accuracy
29	7.81	0.904 ± 0.059
15	4.85	0.904 ± 0.067
10	3.76	0.908 ± 0.058
6	2.92	0.888 ± 0.065
3	1.99	0.883 ± 0.069
2	1.87	0.858 ± 0.060

5 Discussion

By introducing diversity curves as size-aware graph representations, our work contributes towards a unified framework for graph structural comparisons through coarsening. Diversity curves provably improve expressivity through edge contraction pooling and exhibit top performance in geometric tasks and unsupervised settings. Nevertheless, our approach has certain *limitations*: We focus on structural comparisons rather than attributed graphs and do not aim to reach state-of-the-art performance in supervised settings. Instead, diversity curves offer interpretable and expressive tools for *unsupervised* structural and distributional comparisons between graphs. Moreover, our methods implicitly rely on the quality and utility of the selected graph coarsening procedure. For example, some of our theoretical results assume we compute diversity curves exactly for all possible choices of edge contractions, which may not always be feasible. Building upon our work, there is potential to generalise the choice of coarsening process, or the graph invariant to track over the coarsening process. In addition, there is the possibility to change the underlying metric for diversity computations as discussed in Section B.1.2 and used in Section 4.5. Finally, diversity curves could be integrated into supervised graph learning architectures as expressive structural encodings.

Acknowledgments and Disclosure of Funding

This work has received funding from the Swiss State Secretariat for Education, Research, and Innovation (SERI). G. W. was partially funded by Humboldt Research Fellowship, CIFAR AI Chair, NSERC Discovery grant 03267, FRQNT grant 343567, and NSF grant DMS-2327211. The content provided here is solely the responsibility of the authors and does not necessarily represent the views of the funding agencies. The authors declare *no* competing interests. The funders had no role in the preparation of the manuscript or the decision to publish.

References

[1] L. Babai and E. M. Luks. Canonical labeling of graphs. In *Proceedings of the Fifteenth Annual ACM Symposium on Theory of Computing*, STOC '83, pages 171–183. Association for Computing Machinery, 1983.

[2] R. Ballester and B. Rieck. On the expressivity of persistent homology in graph learning. In *Proceedings of the Third Learning on Graphs Conference*, pages 42–1. PMLR, 2025.

[3] R. Ballester, E. Röell, D. B. Schmid, M. Alain, S. Escalera, C. Casacuberta, and B. Rieck. MANTRA: The manifold triangulations assemblage. In *International Conference on Learning Representations*, 2025.

[4] P. Berman and S. P. Kasiviswanathan. Faster approximation of distances in graphs. In F. Dehne, J.-R. Sack, and N. Zeh, editors, *Algorithms and Data Structures*, pages 541–552, 2007.

[5] F. H. Biase, X. Cao, and S. Zhong. Cell fate inclination within 2-cell and 4-cell mouse embryos revealed by single-cell RNA sequencing. *Genome research*, 24(11):1787–1796, 2014.

[6] K. M. Borgwardt, C. S. Ong, S. Schönauer, S. V. N. Vishwanathan, A. J. Smola, and H.-P. Kriegel. Protein function prediction via graph kernels. *Bioinformatics*, 21(Suppl 1):i47–i56, 2005.

[7] F. Buettner, K. N. Natarajan, F. P. Casale, V. Proserpio, A. Scialdone, F. J. Theis, S. A. Teichmann, J. C. Marioni, and O. Stegle. Computational analysis of cell-to-cell heterogeneity in single-cell RNA-sequencing data reveals hidden subpopulations of cells. *Nature biotechnology*, 33(2):155–160, 2015.

[8] J. C. Burns, M. C. Kelly, M. Hoa, R. J. Morell, and M. W. Kelley. Single-cell RNA-seq resolves cellular complexity in sensory organs from the neonatal inner ear. *Nature communications*, 6 (1):8557, 2015.

[9] J. G. Camp, F. Badsha, M. Florio, S. Kanton, T. Gerber, M. Wilsch-Bräuninger, E. Lewitus, A. Sykes, W. Hevers, M. Lancaster, et al. Human cerebral organoids recapitulate gene expression programs of fetal neocortex development. *Proceedings of the National Academy of Sciences*, 112(51):15672–15677, 2015.

[10] F. Chazal, M. Glisse, C. Labruère, and B. Michel. Convergence rates for persistence diagram estimation in topological data analysis. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32, pages 163–171, 2014.

[11] L.-F. Chu, N. Leng, J. Zhang, Z. Hou, D. Mamott, D. T. Vereide, J. Choi, C. Kendziorski, R. Stewart, and J. A. Thomson. Single-cell RNA-seq reveals novel regulators of human embryonic stem cell differentiation to definitive endoderm. *Genome biology*, 17(1):173, 2016.

[12] J. L. Close, Z. Yao, B. P. Levi, J. A. Miller, T. E. Bakken, V. Menon, J. T. Ting, A. Wall, A.-R. Krostag, E. R. Thomsen, et al. Single-cell profiling of an in vitro model of human interneuron development reveals temporal dynamics of cell type production and maturation. *Neuron*, 93(5): 1035–1048, 2017.

[13] T. T. S. Consortium*, R. C. Jones, J. Karkanias, M. A. Krasnow, A. O. Pisco, S. R. Quake, J. Salzman, N. Yosef, B. Bulthaup, P. Brown, et al. The tabula sapiens: A multiple-organ, single-cell transcriptomic atlas of humans. *Science*, 376(6594):eabl4896, 2022.

[14] C. Coupette, J. Wayland, E. Simons, and B. Rieck. No metric to rule them all: Toward principled evaluations of graph-learning datasets. In *Proceedings of the 42nd International Conference on Machine Learning*, volume 267, pages 11405–11434. PMLR, 2025.

[15] S. Darmanis, S. A. Sloan, Y. Zhang, M. Enge, C. Caneda, L. M. Shuer, M. G. Hayden Gephart, B. A. Barres, and S. R. Quake. A survey of human brain transcriptome diversity at the single cell level. *Proceedings of the National Academy of Sciences*, 112(23):7285–7290, 2015.

[16] Q. Deng, D. Ramsköld, B. Reinius, and R. Sandberg. Single-cell RNA-seq reveals dynamic, random monoallelic gene expression in mammalian cells. *Science*, 343(6167):193–196, 2014.

[17] P. D. Dobson and A. J. Doig. Distinguishing enzyme structures from non-enzymes without alignments. *Journal of Molecular Biology*, 330(4):771–783, 2003.

[18] M. Dory, S. Forster, Y. Kirkpatrick, Y. Nazari, V. V. Williams, and T. d. Vos. Fast 2-approximate all-pairs shortest paths. In *Proceedings of the 2024 Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, pages 4728–4757. SIAM, 2024.

[19] K. Dunne. Efficiently approximating spread dimension with high confidence. *arXiv preprint arXiv:2408.14590*, 2024.

[20] I. Engel, G. Seumois, L. Chavez, D. Samaniego-Castruita, B. White, A. Chawla, D. Mock, P. Vijayanand, and M. Kronenberg. Innate-like functions of natural killer T cell subsets result from highly divergent gene programs. *Nature immunology*, 17(6):728–739, 2016.

[21] S. Frazer, J. Prados, M. Niquille, C. Cadilhac, F. Markopoulos, L. Gomez, U. Tomasello, L. Telley, A. Holtmaat, D. Jabaudon, et al. Transcriptomic and anatomic parcellation of 5-HT3AR expressing cortical interneuron subtypes revealed by single-cell RNA sequencing. *Nature communications*, 8(1):14219, 2017.

[22] O. Gokce, G. M. Stanley, B. Treutlein, N. F. Neff, J. G. Camp, R. C. Malenka, P. E. Rothwell, M. V. Fuccillo, T. C. Südhof, and S. R. Quake. Cellular taxonomy of the mouse striatum as revealed by single-cell RNA-seq. *Cell reports*, 16(4):1126–1137, 2016.

[23] M. Goolam, A. Scialdone, S. J. Graham, I. C. Macaulay, A. Jedrusik, A. Hupalowska, T. Voet, J. C. Marioni, and M. Zernicka-Goetz. Heterogeneity in Oct4 and Sox2 targets biases cell fate in 4-cell mouse embryos. *Cell*, 165(1):61–74, 2016.

[24] R. Z. Granit, H. Masury, R. Condiotti, Y. Fixler, Y. Gabai, T. Glikman, S. Dalin, E. Winter, Y. Nevo, E. Carmon, et al. Regulation of cellular heterogeneity and rates of symmetric and asymmetric divisions in triple-negative breast cancer. *Cell reports*, 24(12):3237–3250, 2018.

[25] D. Grattarola, D. Zambon, F. M. Bianchi, and C. Alippi. Understanding pooling in graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 35(2):2708–2718, 2022.

[26] F. Guo, L. Yan, H. Guo, L. Li, B. Hu, Y. Zhao, J. Yong, Y. Hu, X. Wang, Y. Wei, et al. The transcriptome and DNA methylome landscapes of human primordial germ cells. *Cell*, 161(6): 1437–1452, 2015.

[27] N. Habib, Y. Li, M. Heidenreich, L. Swiech, I. Avraham-Davidi, J. J. Trombetta, C. Hession, F. Zhang, and A. Regev. Div-seq: Single-nucleus RNA-seq reveals dynamics of rare adult newborn neurons. *Science*, 353(6302):925–928, 2016.

[28] A. A. Hagberg, D. A. Schult, and P. J. Swart. Exploring network structure, dynamics, and function using networkx. In *Proceedings of the 7th Python in Science Conference*, pages 11 – 15, 2008.

[29] R. Han, G. Yao, M. Li, W. Zhang, Y. Li, W. Xin, H. Lei, M. Li, Z. Zhang, C. Du, et al. Multi-relation graph-kernel strengthen network for graph-level clustering. In *2025 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2025.

[30] X. Han, R. Wang, Y. Zhou, L. Fei, H. Sun, S. Lai, A. Saadatpour, Z. Zhou, H. Chen, F. Ye, et al. Mapping the mouse cell atlas by microwell-seq. *Cell*, 172(5):1091–1107, 2018.

[31] T. Hayashi, H. Ozaki, Y. Sasagawa, M. Umeda, H. Danno, and I. Nikaido. Single-cell full-length total RNA sequencing uncovers dynamics of recursive splicing and enhancer RNAs. *Nature communications*, 9(1):619, 2018.

[32] B. Hie, H. Cho, B. DeMeo, B. Bryson, and B. Berger. Geometric sketching compactly summarizes the single-cell transcriptomic landscape. *Cell systems*, 8(6):483–493, 2019.

[33] H. Hochgerner, A. Zeisel, P. Lönnerberg, and S. Linnarsson. Conserved properties of dentate gyrus neurogenesis across postnatal development revealed by single-cell RNA sequencing. *Nature neuroscience*, 21(2):290–299, 2018.

[34] T. Huang, Z. Hu, and R. Ying. Learning to group auxiliary datasets for molecule. *Advances in Neural Information Processing Systems*, 36:45339–45359, 2023.

[35] Y. Jang, S. Lee, and S. Ahn. A simple and scalable representation for graph generation. In *International Conference on Learning Representations*, 2024.

[36] S. Jin and R. Zafarani. The spectral zoo of networks: Embedding and visualizing networks with spectral moments. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1426–1434, 2020.

[37] S. Joost, A. Zeisel, T. Jacob, X. Sun, G. La Manno, P. Lönnerberg, S. Linnarsson, and M. Kasper. Single-cell transcriptomics reveals that differentiation and spatial signatures shape epidermal and hair follicle heterogeneity. *Cell systems*, 3(3):221–237, 2016.

[38] C. K. Joshi, Q. Cappart, L.-M. Rousseau, and T. Laurent. Learning the travelling salesperson problem requires rethinking generalization. *Constraints*, 27:70–98, 2022.

[39] T. Kim, I. R. Chen, Y. Lin, A. Y.-Y. Wang, J. Y. H. Yang, and P. Yang. Impact of similarity metrics on single-cell RNA-seq data clustering. *Briefings in bioinformatics*, 20(6):2316–2326, 2019.

[40] A. M. Klein, L. Mazutis, I. Akartuna, N. Tallapragada, A. Veres, V. Li, L. Peshkin, D. A. Weitz, and M. W. Kirschner. Droplet barcoding for single-cell transcriptomics applied to embryonic stem cells. *Cell*, 161(5):1187–1201, 2015.

[41] C. Koke, Y. Shen, A. Saroha, M. Eisenberger, B. Rieck, M. Bronstein, and D. Cremers. Graph neural networks are not continuous across graph resolutions. In *Proceedings of the 43rd International Conference on Machine Learning*, Proceedings of Machine Learning Research. PMLR, 2026.

[42] M. S. Kowalczyk, I. Tirosh, D. Heckl, T. N. Rao, A. Dixit, B. J. Haas, R. K. Schneider, A. J. Wagers, B. L. Ebert, and A. Regev. Single-cell RNA-seq reveals changes in cell cycle and differentiation programs upon aging of hematopoietic stem cells. *Genome research*, 25(12): 1860–1872, 2015.

[43] M. Krahn and V. Garg. Heat kernel goes topological. *arXiv preprint arXiv:2507.12380*, 2025.

[44] M. Krimmel, P. Hartout, K. Borgwardt, and D. Chen. Polygraph discrepancy: a classifier-based metric for graph generation. In *International Conference on Learning Representations*, 2026.

[45] M. Krzak, Y. Raykov, A. Boukouvalas, L. Cutillo, and C. Angelini. Benchmark and parameter sensitivity analysis of single-cell RNA sequencing clustering methods. *Frontiers in genetics*, 10:1253, 2019.

[46] V. Lachi, A. Moallemy-Oureh, A. Roth, and P. Welke. Expressive pooling for graph neural networks. *Transactions on Machine Learning Research*, 2025.

[47] T. Leinster. The magnitude of metric spaces. *Documenta Mathematica*, 18:857–905, 2013.

[48] N. Leng, L.-F. Chu, C. Barry, Y. Li, J. Choi, X. Li, P. Jiang, R. M. Stewart, J. A. Thomson, and C. Kendziorski. Oscope identifies oscillatory genes in unsynchronized single-cell RNA-seq experiments. *Nature methods*, 12(10):947–950, 2015.

[49] H. Li, E. T. Courtois, D. Sengupta, Y. Tan, K. H. Chen, J. J. L. Goh, S. L. Kong, C. Chua, L. K. Hon, W. S. Tan, et al. Reference component analysis of single-cell transcriptomes elucidates cellular heterogeneity in human colorectal tumors. *Nature genetics*, 49(5):708–718, 2017.

[50] H. Li, F. Horns, B. Wu, Q. Xie, J. Li, T. Li, D. J. Luginbuhl, S. R. Quake, and L. Luo. Classifying drosophila olfactory projection neuron subtypes by single-cell RNA sequencing. *Cell*, 171(5): 1206–1220, 2017.

[51] L. Li, J. Dong, L. Yan, J. Yong, X. Liu, Y. Hu, X. Fan, X. Wu, H. Guo, X. Wang, et al. Single-cell RNA-seq analysis maps development of human germline cells and gonadal niche interactions. *Cell stem cell*, 20(6):858–873, 2017.

[52] H. S. Lim and P. Qiu. Quantifying the clusterness and trajectoriness of single-cell RNA-seq data. *PLoS Computational Biology*, 20(2):e1011866, 2024.

[53] K. Limbeck, R. Andreeva, R. Sarkar, and B. Rieck. Metric space magnitude for evaluating the diversity of latent representations. In *Advances in Neural Information Processing Systems*, volume 37, 2024.

[54] K. Limbeck, L. Mezrag, G. Wolf, and B. Rieck. Geometry-aware edge pooling for graph neural networks. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[55] C. Liu, Y. Zhan, J. Wu, C. Li, B. Du, W. Hu, T. Liu, and D. Tao. Graph pooling for graph neural networks: Progress, challenges, and opportunities. In *Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence, IJCAI-23*, pages 6712–6722, 2023.

[56] K. M. Loh, A. Chen, P. W. Koh, T. Z. Deng, R. Sinha, J. M. Tsai, A. A. Barkal, K. Y. Shen, R. Jain, R. M. Morganti, et al. Mapping the pairwise choices leading from pluripotency to human bone, heart, and other mesoderm cell types. *Cell*, 166(2):451–467, 2016.

[57] S. Marques, A. Zeisel, S. Codeluppi, D. Van Bruggen, A. Mendanha Falcão, L. Xiao, H. Li, M. Häring, H. Hochgerner, R. A. Romanov, et al. Oligodendrocyte heterogeneity in the mouse juvenile and adult central nervous system. *Science*, 352(6291):1326–1329, 2016.

[58] K. Martinkus, A. Loukas, N. Perraudin, and R. Wattenhofer. Spectre: Spectral conditioning helps to overcome the expressivity limits of one-shot graph generators. In *Proceedings of the 39th International Conference on Machine Learning*, pages 15159–15179. PMLR, 2022.

[59] C. Morris, N. M. Kriege, F. Bause, K. Kersting, P. Mutzel, and M. Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. In *ICML 2020 Workshop on Graph Representation Learning and Beyond (GRL+ 2020)*, 2020.

[60] T. Nakamura, I. Okamoto, K. Sasaki, Y. Yabuta, C. Iwatani, H. Tsuchiya, Y. Seita, S. Nakamura, T. Yamamoto, and M. Saitou. A developmental coordinate of pluripotency among mice, monkeys and humans. *Nature*, 537(7618):57–62, 2016.

[61] T. Nakamura, Y. Yabuta, I. Okamoto, K. Sasaki, C. Iwatani, H. Tsuchiya, and M. Saitou. Single-cell transcriptome of early embryos and cultured embryonic stem cells of cynomolgus monkeys. *Scientific data*, 4(1):170067, 2017.

[62] A. Narayanan, M. Chandramohan, R. Venkatesan, L. Chen, Y. Liu, and S. Jaiswal. graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*, 2017.

[63] L. O'Bray, B. Rieck, and K. Borgwardt. Filtration curves for graph representation. In *Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 1267–1275, New York, NY, USA, 2021. Association for Computing Machinery.

[64] C. Ollivier-Gooch. Coarsening unstructured meshes by edge contraction. *International Journal for Numerical Methods in Engineering*, 57(3):391–414, 2003.

[65] A. Olsson, M. Venkatasubramanian, V. K. Chaudhri, B. J. Aronow, N. Salomonis, H. Singh, and H. L. Grimes. Single-cell analysis of mixed-lineage states leading to a binary cell fate choice. *Nature*, 537(7622):698–702, 2016.

[66] L. O'Bray, M. Horn, B. A. Rieck, and K. Borgwardt. Evaluation metrics for graph generative models: Problems, pitfalls, and practical solutions. In *International Conference on Learning Representations*, 2022.

[67] J. Park, R. Shrestha, C. Qiu, A. Kondo, S. Huang, M. Werth, M. Li, J. Barasch, and K. Suszták. Single-cell transcriptomics of the mouse kidney reveals potential cellular targets of kidney disease. *Science*, 360(6390):758–763, 2018.

[68] S. Petropoulos, D. Edsgärd, B. Reinius, Q. Deng, S. P. Panula, S. Codeluppi, A. P. Reyes, S. Linnarsson, R. Sandberg, and F. Lanner. Single-cell RNA-seq reveals lineage and x chromosome dynamics in human preimplantation embryos. *Cell*, 165(4):1012–1026, 2016.

[69] B. Phipson and G. Smyth. Permutation p-values should never be zero: calculating exact p-values when permutations are randomly drawn. *Statistical Applications in Genetics and Molecular Biology*, 9:Article39–Article39, 2010.

[70] M. Plass, J. Solana, F. A. Wolf, S. Ayoub, A. Misios, P. Glažar, B. Obermayer, F. J. Theis, C. Kocks, and N. Rajewsky. Cell type atlas and lineage tree of a whole complex animal by single-cell transcriptomics. *Science*, 360(6391):eaaq1723, 2018.

[71] A. Poirier. *Reconstructing a graph from its edge-contractions*. PhD thesis, Université d'Ottawa/University of Ottawa, 2018.

[72] W.-L. Qiu, Y.-W. Zhang, Y. Feng, L.-C. Li, L. Yang, and C.-R. Xu. Deciphering pancreatic islet β cell and α cell maturation pathways and characteristic features at the single-cell level. *Cell metabolism*, 25(5):1194–1205, 2017.

[73] E. Roff and M. Yoshinaga. The small-scale limit of magnitude and the one-point property. *Bulletin of the London Mathematical Society*, 57(6):1841–1855, 2025.

[74] B. Rozemberczki and R. 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 & knowledge management*, pages 1325–1334, 2020.

[75] B. Rozemberczki, O. Kiss, and R. Sarkar. Karate club: An API oriented open-source python framework for unsupervised learning on graphs. In *Proceedings of the 29th ACM International Conference on Information and Knowledge Management (CIKM '20)*, pages 3125—-3132, 2020.

[76] W. Saelens, R. Cannoodt, H. Todorov, and Y. Saeys. A comparison of single-cell trajectory inference methods. *Nature biotechnology*, 37(5):547–554, 2019.

[77] A.-E. Saliba, L. Li, A. J. Westermann, S. Appenzeller, D. A. Stapels, L. N. Schulte, S. Helaine, and J. Vogel. Single-cell RNA-seq ties macrophage polarization to growth rate of intracellular salmonella. *Nature microbiology*, 2(2):16206, 2016.

[78] A. K. Shalek, R. Satija, J. Shuga, J. J. Trombetta, D. Gennert, D. Lu, P. Chen, R. S. Gertner, J. T. Gaublomme, N. Yosef, et al. Single-cell RNA-seq reveals dynamic paracrine control of cellular variation. *Nature*, 510(7505):363–369, 2014.

[79] G. Siglidis, G. Nikolentzos, S. Limnios, C. Giatsidis, K. Skianis, and M. Vazirgiannis. GraKeL: A graph kernel library in python. *Journal of Machine Learning Research*, 21(54):1–5, 2020.

[80] C. Siu, G. Samorodnitsky, C. L. Yu, and R. He. The asymptotics of the expected betti numbers of preferential attachment clique complexes. *Advances in Applied Probability*, 57(3):940–968, 2025.

[81] J. Southern, J. Wayland, M. Bronstein, and B. Rieck. Curvature filtrations for graph generative model evaluation. *Advances in Neural Information Processing Systems*, 36:63036–63061, 2023.

[82] M. Tantardini, F. Ieva, L. Tajoli, and C. Piccardi. Comparing methods for comparing networks. *Scientific reports*, 9(1):17557, 2019.

[83] B. Tasic, V. Menon, T. N. Nguyen, T. K. Kim, T. Jarsky, Z. Yao, B. Levi, L. T. Gray, S. A. Sorensen, T. Dolbeare, et al. Adult mouse cortical cell taxonomy revealed by single cell transcriptomics. *Nature neuroscience*, 19(2):335–346, 2016.

[84] L. Tian, X. Dong, S. Freytag, K.-A. Lê Cao, S. Su, A. JalalAbadi, D. Amann-Zalcenstein, T. S. Weber, A. Seidi, J. S. Jabbari, et al. Benchmarking single cell rna-sequencing analysis pipelines using mixture control experiments. *Nature methods*, 16(6):479–487, 2019.

[85] I. Tirosh, B. Izar, S. M. Prakadan, M. H. Wadsworth, D. Treacy, J. J. Trombetta, A. Rotem, C. Rodman, C. Lian, G. Murphy, et al. Dissecting the multicellular ecosystem of metastatic melanoma by single-cell RNA-seq. *Science*, 352(6282):189–196, 2016.

[86] A. Tola, F. M. Taiwo, C. G. Akcora, and B. Coskunuzer. TopER: Topological embeddings in graph representation learning. In *The Thirty-ninth Annual Conference on Neural Information Processing Systems*, 2025.

[87] C. Trapnell, D. Cacchiarelli, J. Grimsby, P. Pokharel, S. Li, M. Morse, N. J. Lennon, K. J. Livak, T. S. Mikkelsen, and J. L. Rinn. The dynamics and regulators of cell fate decisions are revealed by pseudotemporal ordering of single cells. *Nature biotechnology*, 32(4):381–386, 2014.

[88] B. Treutlein, D. G. Brownfield, A. R. Wu, N. F. Neff, G. L. Mantalas, F. H. Espinoza, T. J. Desai, M. A. Krasnow, and S. R. Quake. Reconstructing lineage hierarchies of the distal lung epithelium using single-cell RNA-seq. *Nature*, 509(7500):371–375, 2014.

[89] B. Treutlein, Q. Y. Lee, J. G. Camp, M. Mall, W. Koh, S. A. M. Shariati, S. Sim, N. F. Neff, J. M. Skotheim, M. Wernig, et al. Dissecting direct reprogramming from fibroblast to neuron using single-cell RNA-seq. *Nature*, 534(7607):391–395, 2016.

[90] A. Tsitsulin, D. Mottin, P. Karras, A. Bronstein, and E. Müller. NetLSD: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 2347–2356, 2018.

[91] A.-C. Villani, R. Satija, G. Reynolds, S. Sarkizova, K. Shekhar, J. Fletcher, M. Griesbeck, A. Butler, S. Zheng, S. Lazo, et al. Single-cell RNA-seq reveals new types of human blood dendritic cells, monocytes, and progenitors. *Science*, 356(6335):eaah4573, 2017.

[92] Y. Wang and M. Zhang. An empirical study of realized GNN expressiveness. In *Proceedings of the 41st International Conference on Machine Learning*, volume 235, pages 52134–52155. PMLR, 2024.

[93] S. Willerton. Spread: a measure of the size of metric spaces. *International Journal of Computational Geometry & Applications*, 25(03):207–225, 2015.

[94] Y. Xin, J. Kim, H. Okamoto, M. Ni, Y. Wei, C. Adler, A. J. Murphy, G. D. Yancopoulos, C. Lin, and J. Gromada. RNA sequencing of single human islet cells reveals type 2 diabetes genes. *Cell metabolism*, 24(4):608–615, 2016.

[95] L. Yang, W.-H. Wang, W.-L. Qiu, Z. Guo, E. Bi, and C.-R. Xu. A single-cell transcriptomic analysis reveals precise pathways and regulatory mechanisms underlying hepatoblast differentiation. *Hepatology*, 66(5):1387–1401, 2017.

[96] G. Yehudai, E. Fetaya, E. Meirom, G. Chechik, and H. Maron. From local structures to size generalization in graph neural networks. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139, pages 11975–11986. PMLR, 2021.

[97] Y. Yin, Q. Wang, S. Huang, H. Xiong, and X. Zhang. AutoGCL: Automated graph contrastive learning via learnable view generators. *Proceedings of the AAAI Conference on Artificial Intelligence*, 36(8):8892–8900, Jun. 2022.

[98] Y. You, T. Chen, Y. Sui, T. Chen, Z. Wang, and Y. Shen. Graph contrastive learning with augmentations. In *Proceedings of the 34th International Conference on Neural Information Processing Systems*, NIPS '20. Curran Associates Inc., 2020.

[99] A. Zeisel, A. B. Muñoz-Manchado, S. Codeluppi, P. Lönnerberg, G. La Manno, A. Juréus, S. Marques, H. Munguba, L. He, C. Betsholtz, et al. Cell types in the mouse cortex and hippocampus revealed by single-cell RNA-seq. *Science*, 347(6226):1138–1142, 2015.

Appendix (Supplementary Materials)

A Changelog **18**

B Extended methods **19**

B.1 Diversity curves . . . 19

B.1.1 Coarsening operations . . . 19

B.1.2 Metric choices . . . 20

B.1.3 Choice of coarsening levels . . . 22

B.1.4 Algorithm . . . 22

B.2 Compute resources . . . 23

B.3 Baseline methods . . . 23

C Proofs and theoretical results **23**

C.1 Computational complexity . . . 23

C.2 Diversity curves collapse to the same diversity . . . 24

C.3 Diversity curves over all possible edge contractions . . . 25

C.4 Diversity curves as graph invariants . . . 25

C.5 Diversity curves increase expressivity . . . 26

D Extended experiments **28**

D.1 Graph perturbations . . . 28

D.2 Simulated graph distributions . . . 29

D.2.1 Distinguishing graph distributions . . . 29

D.2.2 Distinguishing parameter choices . . . 31

D.3 Distinguishing between trajectory or cluster structures in single-cell graphs . . . 34

D.4 Characterising molecular graph datasets . . . 35

D.5 Distinguishing geometric shapes . . . 36

D.6 Expressivity . . . 38

D.7 Computational efficiency . . . 39

D.8 Ablation studies . . . 41

D.8.1 Choice of distance and edge scores . . . 41

D.8.2 Stability with respect to the edge contraction sequence . . . 42

D.8.3 Choice of evaluation scales, expressivity, and computational cost . . . 43

D.8.4 Performance across up- or downsampling . . . 43

D.9 Extended results on graph classification datasets . . . 44

A Changelog

We provide a brief list of changes we implemented after the reviewing process. Overall, we also improved the text with minor stylistic choices and fixing of typos.

- Changes in the main text:
  - In Section 1, we clarify the contributions of our method.
  - Section 2 has been rewritten with the feedback of the reviewers in mind to strengthen the context and novelty of our work with respect to existing work.
  - We add clarifications in Section 3.1, i.e. we argue what properties are needed in the coarsening procedure and rephrase the formal exposition of the edge contraction sequence to make it more concise.
  - We redefine the definition of a diversity curve in Section 3.1 to be the average over a multiset of graph coarsening sequences, unifying the way we discuss diversity curves. We reformulate Section 3.2 to reflect this change. We further address the discussion about the theoretical and practical evaluation of diversity curves brought up during the review process.
  - In Section 3.1, we remove the paragraph "Applications to graph-level tasks" by moving the key points to "Diversity curves as size-aware graph representation."
  - The theoretical time complexity computation in Section 3.2 has been made more concise.
  - We clarify in Section 3.2 that the setting for Theorem 3 considers graphs of the same cardinality.
  - We discuss the possibility to use parallelisation to increase the computational efficiency in Section 3.2.
  - Section 4.2 clarifies how the silhouette scores have been computed.
  - In Section 4.3, we add comparisons to feature-based baselines in order to assess what information the features only provide.
  - We specify the number of nodes of the graphs for the experiment on geometric shapes in Section 4.5.
  - We have included a short version of the results table for Section 4.6.
  - We add a brief description of extended results and ablation experiments in Section 4.7.
- Changes in the appendix:
  - In Section B.1.1, we give a more detailed discussion on what properties we require in the coarsening operation and why these requirements lead us to choose edge contraction pooling. We detail in Theorem 2 for which theoretical results the requirements are needed.
  - We include a more detailed discussion on the choice of metric in Section B.1.2.
  - We make Section B and Section C consistent with the adapted version of the definition of diversity curves. In Section C, we include a section discussing the new formulation of "the diversity curves over all possible edge contractions" and why it was chosen over the previous definition of "the multiset of diversity curves over all possible edge contractions."
  - We fix some typos in Section C in the proof of Theorem 2.
  - We add structural baseline comparisons for distinguishing simulated graph distributions in Section D.2.
  - In Section D.3, we add comparisons to feature-based baselines in order to assess what information the features only provide.
  - We emphasise in Section D.7 how we can flexibly define the cardinalities at which we compute the diversity in order to compute diversity curves of larger graphs efficiently and add an experiment for distinguishing graph distributions with graphs between cardinalities ranging from 16 to 4096.
  - For the empirical cost in Section D.7, we include a speed-up for the empirical computational complexity of diversity curves.

- We add further ablations to Section D.8 as discussed during the review process. Specifically, we provide empirical stability results of diversity curves with respect to varying the contraction sequence and the size of the set over which we compute the diversity curves as discussed with the reviewers. In Section D.8.3, we add an experiment on the trade-off between the number of evaluation scales, expressiveness, and the computational cost. We include the ablation study of only using up- or only downsampling in Section D.8.4.
- Section D.9 gives additional results on the `MCF-7` and `MOLT-4` datasets.

B Extended methods

To support the introduction of diversity curves in Section 3.1, we give a more detailed description of our proposed methods, as well as the steps involved in computing diversity curves, and the outline of the final algorithm. Further, we briefly discuss the compute resources used for our experiments, and give a brief introduction to unsupervised baseline methods, with which we compare diversity curves throughout our main experiments.

B.1 Diversity curves

In the following, we define how to compute diversity curves in detail, describe the algorithm used to compute them, and discuss specific parameter choices, such as the choice of pooling operation and the choice of graph metric.

B.1.1 Coarsening operations

Throughout our work, we explore edge contraction pooling to downsample the size of graphs. Further, we use upsampling operations through upsampling nodes in a graph or triangles in a mesh. These operations are detailed below. The coarsening operator used to compute diversity curves should satisfy the following conditions, previously summarised in Section 3.1:

i) The coarsening operator is a function $\mathrm{Coarse}(G, i)$ that takes as inputs a graph G with c connected components and cardinality n and any integer i such that $c \leq i \leq n$ and outputs a graph with cardinality $|\mathrm{Coarse}(G, i)| = i$.

ii) The coarsening is iterative. That is for any $i < j < |G|$ we can obtain the coarsening to i by first coarsening to j, concretely $\mathrm{Coarse}(G, i) = \mathrm{Coarse}(\mathrm{Coarse}(G, j), i)$.

iii) The coarsening is applicable to unattributed graphs.

iv) The coarsening is unsupervised.

v) The coarsening process does not change the connected components of the graph. That is, for two nodes in a graph G of the same connected component, their respective supernodes in the pooled graph belong to the same connected component.

Note that these requirements allow us to consider specific sequences of sequentially coarsened graphs and control precisely the cardinality in order to evaluate the diversity curves. Edge contraction coarsening as described in more detail below has been studied in previous works [54] and satisfies all of the conditions above. In contrast, many other coarsening operations on graphs fail to meet these requirements. Pooling layers as for example discussed by Grattarola et al. [25], might either not directly induce a stepwise sequence of coarsened graphs, or even if they produce a stepwise coarsening hierarchy not enable precise control over the number of nodes in the coarsened graphs. Methods that cluster nodes, such as Graclus or NMF, do not directly allow us to iteratively pool graphs to any desired size (requirement i and ii). Other operations that delete nodes, such as TopK or SAGPool, might disconnect the graph (requirement v). Further, we explicitly choose not to apply supervised pooling methods (requirement iv), excluding a wide range of trainable operations, such as TopK, SAGPool, DiffPool or MinCutPool. Hence, considering related work on graph pooling and graph coarsening methods, we do find that the iterative edge-coarsening approach detailed below is particularly suitable for the purpose of our work.

Edge pooling We downsample graphs by performing edge contractions closely following the pooling method proposed by Limbeck et al. [54]. For a graph $G = (X, E)$ and an edge $e = (u, v) \in E$, we define the graph G/e obtained from G by contracting the edge e to have vertex set $X \setminus \{v\}$ and edge set $E \setminus \{(v, x) | x \in \mathcal{N}_G(v)\} \cup \{(u, x) | x \in \mathcal{N}_G(v)\}$, where $\mathcal{N}_G(v) = \{x \in G | (v, x) \in E\}$ denotes the neighbourhood of the vertex v in G. Note that we do not introduce any multiple edges or a self-loop on the remaining node u. To describe our downsampling process, let $G = (X, E)$ be a graph of size $n = |X|$ with $m = |E|$ edges that we want to downsample to the size $n - k$, where $0 < k \leq m$. To do this, we generate a sequence $\mathcal{E} = (e_i)_{1 \leq i \leq k}$ of edges in G to be contracted in that order. We choose this edge sequence based on edge scores that are either drawn uniformly at random from the unit interval for each edge, or we use edge scores based on a diversity measure as proposed by Limbeck et al. [54]. In the latter case, the score for an edge $e \in E$ is the difference in diversity $|\operatorname{Div}(G) - \operatorname{Div}(G/e)|$. After the scores are computed, we iteratively add the edge with the lowest score to the sequence $\mathcal{E}$ if that edge is not adjacent to a previously contracted edge. If there are no more such edges, we recompute all edge scores and repeat the procedure. With the generated edge sequence, we define the sequence of coarsened graphs $(G_i)_{n-k \leq i \leq n}$ where $G_n = G$ and for $n - k < i \leq n$ we set $G_{i-1} = G_i / e_{n-i+1}$. If node features are available, we average the node features of all original nodes that have been merged through these edge contractions. Further, we note that edge contraction pooling extends to mesh data and higher order shapes and is in fact often used to coarsen mesh data in practice [64]. Hence, we also apply edge contraction pooling to the triangulations in Section 4.5 while taking care to track triangles as well as edges through the coarsening process. Edge contraction pooling is thus used throughout all of our experiments.

Node upsampling For upsampling a graph $G = (X, E)$ of size $n = |X|$ to size $N > n$, we generate a sequence $\mathcal{V} = (v_i)_{1 \leq i \leq N-n}$ of $N - n$ nodes from G. This sequence determines the nodes that we upsample and the order in which we upsample them. We construct the sequence in such a manner that the difference between the number of times nodes are selected is no more than 1. To do this, let $m, r \in \mathbb{Z}_{\geq 0}$ be such that $N - n = m \cdot n + r$ and choose the first $m \cdot n$ nodes in $\mathcal{V}$ by doing m permutations of all nodes in the graph. For the remaining r nodes we choose them from the set of nodes X without replacement. This ensures each node is picked at least once before a previously upsampled node is chosen again. Going through the sequence $\mathcal{V}$ of nodes to upsample, for every $0 \leq i \leq N - n$ we add a new node u with the same neighbourhood as the sampled node v_i and also connect the sampled node to the new node. That is, we add an edge between the new node u and every node in $\mathcal{N}_G(v_i) \cup \{v_i\}$. If node features are available, we set the node feature of u to be the same as the node feature of v_i. This node-based upsampling procedure is used throughout all of our experiments other than in Section 4.5. Note that the node-based upsampling strategy aims to mimic and reverse the way edge contraction pooling downsamples nodes in a graph.

Triangular upsampling Given a triangular mesh, such as in Section 4.5, we might not want to just upsample nodes in the graphs as this loses information on the inherent shape of the data. Instead, we employ triangular upsampling and upsample each 3-clique in the graph, or each triangle in the triangulation. Specifically, we uniformly select triangles making sure that each triangle is picked at least once before a previously upsampled triangle could be selected again. We then subdivide the selected triangles in order by adding a central vertex and connecting it to all corners. This way, we uniformly subdivide the surface of the meshes ($\mathcal{T}$), or the 3-cliques in the graphs (G). This upsampling strategy is used to compute diversity curves in Section 4.5 and account for the triangular nature of the data.

B.1.2 Metric choices

To compute diversity curves, we consider the structural diversity of a graph to be dependent on a metric between nodes. Here, we will define and detail the distance choices explored in our paper. First, we define shortest-path distances, which are our default choice across this paper, as they are faster to compute than other structure-based metrics, such as heat kernel or diffusion distances, while performing well across our experiments and ablation studies.

Definition 2 (Shortest-path distance)**.** *Given an undirected graph $G = (X, E)$, the* shortest-path distance d *is a metric that counts the minimum number of edges in any path between two nodes, $x_i, x_j \in X$, so that $d(x_i, x_j) = \infty$ if no such path exists and otherwise*

$$d(x_i, x_j) := \min_k \{k \in \mathbb{N} | \exists \text{ path } x_i = v_0, v_1, \ldots, v_k = x_j \text{ where } (v_l, v_{l+1}) \in E \text{ and } v_l \in X\}. \quad (3)$$

When dealing with attributed graphs we can further define distances based on node features:

Definition 3 (Euclidean distance)**.** *Given a graph $G = (X, E)$ associated with features $\mathbf{Y} \in \mathbb{R}^{|X| \times M}$ the* Euclidean distance *d between two nodes $x_i, x_j \in X$ with feature vectors $\mathbf{y_i}, \mathbf{y_j} \in \mathbb{R}^M$ is computed as*

$$d(x_i, x_j) := \|\mathbf{y_i} - \mathbf{y_j}\|_2 = \sqrt{\sum_{k=1}^{M} (\mathbf{Y}_{ik} - \mathbf{Y}_{jk})^2}. \tag{4}$$

Next, we define alternative structure-based metrics, which are computed from a graph's normalised Laplacian and have been used throughout related literature to study the geometry of graphs [14, 54]. Let $G = (X, E)$ be a graph, A its adjacency matrix, and D its degree matrix, whose diagonal entries are given by $D_{ii} = \sum_{j=1}^{|X|} A_{ij}$. The normalised graph Laplacian of G is given by $\hat{L} = D^{-\frac{1}{2}}(D - A)D^{-\frac{1}{2}}$.

Definition 4 (Diffusion distance)**.** *Given a graph $G = (X, E)$, its normalised Laplacian, $\hat{L}$, has positive eigenvalues $2 = \lambda_0 > \lambda_1 > \lambda_2 > \cdots > \lambda_{n-1} \geq 0$ with eigenvectors $\{\psi_l\}_l$. At time t this provides an embedding in Euclidean space given by*

$$\Phi_t(x) = (\lambda_1^t \psi_1(x), \cdots, \lambda_{n-1}^t \psi_{n-1}(x)) \text{ for } x \in X. \tag{5}$$

The diffusion distance *d between two nodes $x_i, x_j \in X$ at time t is defined as*

$$d(x_i, x_j) := \|\Phi_t(x_i) - \Phi_t(x_j)\|_2. \tag{6}$$

Definition 5 (Heat kernel distance)**.** *Given a graph $G = (X, E)$, its normalised Laplacian, $\hat{L}$, has positive eigenvalues $2 = \lambda_0 > \lambda_1 > \lambda_2 > \cdots > \lambda_{n-1} \geq 0$ with eigenvectors $\{\psi_l\}_l$. The* heat kernel distance *d between two nodes $x_i, x_j \in X$ at time t is defined as*

$$d(x_i, x_j) := \sum_{l=0}^{n-1} \exp(-\lambda_l t \psi_l(x_i) \psi_l(x_j)). \tag{7}$$

Unless otherwise stated, we set the time parameter to $t = 1$ throughout our experiments. Further, we define a generalisation of the heat kernel distances to higher-dimensional shapes using the Hodge Laplacian on a simplicial complex [43]. We use existing code for computing the Hodge Laplacian.[1]

Definition 6 (Hodge heat kernel distance)**.** *Given a simplicial complex $\mathcal{C}$, its k-th Hodge Laplacian, $\hat{L}$, has eigenvalues $\lambda_0 > \lambda_1 > \lambda_2 > \cdots > \lambda_{s-1}$ with eigenvectors $\{\psi_l\}_l$. We define the* topological heat kernel distance *d between two k-simplices $c_i, c_j \in \mathcal{C}$ at time t as*

$$d(c_i, c_j) := \sum_{l=0}^{s-1} \exp(-\lambda_l t \psi_l(c_i) \psi_l(c_j)). \tag{8}$$

Different choices of metric allow us to explore different notions of geometry and structural diversity on graphs. In practice, we evaluate this choice of metrics across multiple experiments. We choose shortest-path distances as default choices due to their good empirical performance at distinguishing simulated graph distributions (Figure S.10) and scalability compared to eigenvalue-based distances (Section C.1). In other applications, we find that a combination of shortest-path and feature-based Euclidean distances performs best at distinguishing the geometry of single-cell graphs (Table S.7), that a combination of shortest-path and higher-order heat kernel distances can distinguish meshes (Table 4), or that diffusion-based distances can be more expressive than shortest-path distances despite being more computationally expensive (Table S.10). Overall, while there are advantages and disadvantages to each distance choice, the shortest-path metric gives a reasonable and generalist summary of graph structure, while alternative choices can reveal different geometries on the graph. In practice, one can select the default shortest-path metric to make general structural comparisons across graphs, explore application-specific distance choices, or automatically learn the distance choices that perform best for a specific application.

[1] https://github.com/tsitsvero/hodgelaplacians available under an MIT licence.

B.1.3 Choice of coarsening levels

Let $\mathcal{D}$ be a set of graphs for which we want to compute the diversity curves and compare them between each other. We select a set $\mathcal{I}$ of integers, which will represent the cardinalities at which we evaluate the diversity curves. That is, for any graph $G = (X, E) \in \mathcal{D}$ in our dataset we construct a sequence $(G_i)_{i\in\mathcal{I}}$ of graphs with $|G_i| = i$ based on the edge contraction and node upsampling described above and compute the diversity measure for every graph in the sequence $(G_i)_{i\in\mathcal{I}}$ to obtain the diversity curve. By default, we choose the cardinalities $\mathcal{I} = \{1, \dots, \max_{G\in\mathcal{D}} |G|\}$ in our experiments. Note that for a graph G with precisely c connected components, we can not coarsen the graph to less nodes than c by edge contractions alone. In the case where $c > 1$, to still obtain values for every cardinality of the diversity curve, we linearly interpolate the diversity curve between $\mathrm{DivCurve}(G)_1 = 1$ and the lowest $i \in \mathcal{I}$ to which we could downsample the graph by edge contractions. This setup allows for a lot of flexibility, in particular, for datasets in which there are graphs of large cardinalities for which it is not feasible to compute the diversity measure directly, we can choose the maximum cardinality to be much smaller than the largest cardinality in the dataset. Furthermore, we can also choose to evaluate the diversity curves only at certain cardinalities if we are not interested in the finest possible resolution.

B.1.4 Algorithm

We base our computations of structural diversity on `magnipy`[2], a Python library for the computation of metric space diversity measures [53], and `mag_edge_pool`[3], which provides code for computing the spread of a metric space on graphs, and for computing edge contraction pooling operations [54]. For using and manipulating graphs, we use the Python package NetworkX [28] released under a BSD-3-Clause Licence. Then, we implement the computations of diversity curves as described in Section 3.1 via Algorithm 1. A code implementation of our methods is available on GitHub.[4]

Algorithm 1 Diversity Curve Computation

Require: Graph $G = (X, E)$; evaluation scales $\mathcal{I} = \{n_{\min}, \dots, n_{\max}\}$; upsampling method Up; downsampling method Coarse; distance function $d(\cdot,\cdot)$; diversity measure $\mathrm{Div}(\cdot, d)$; number of repetitions $R \in \mathbb{N}$

Ensure: Diversity curve $\mathbf{v} \in \mathbb{R}^{|\mathcal{I}|}$

$\mathbf{v} \leftarrow \mathbf{0}^{|\mathcal{I}|}, \quad n \leftarrow |X|$
for $r = 1, \dots, R$ **do**
  $\mathcal{I}_\uparrow \leftarrow \{i \in \mathcal{I} : i > n\}$ in ascending order
  $\mathcal{I}_\downarrow \leftarrow \{i \in \mathcal{I} : i \le n\}$ in descending order
  $G_\uparrow \leftarrow G$
  for each $i \in \mathcal{I}_\uparrow$ **do**
    $G_\uparrow \leftarrow \mathrm{Up}(G_\uparrow, i)$ {Upsample to i nodes}
    $\mathbf{v}[i] \leftarrow \mathbf{v}[i] + \mathrm{Div}(G_\uparrow, d)$
  end for
  $G_\downarrow \leftarrow G$
  for each $i \in \mathcal{I}_\downarrow$ **do**
    $G_\downarrow \leftarrow \mathrm{Coarse}(G_\downarrow, i)$ {Coarsen to i nodes}
    $\mathbf{v}[i] \leftarrow \mathbf{v}[i] + \mathrm{Div}(G_\downarrow, d)$
  end for
end for
$\mathbf{v} \leftarrow \mathbf{v}/R$
return $\mathbf{v}$

As default settings, we select the following choices: We set the evaluation interval $\mathcal{I} = \{1, 2, \dots, n_{\max}\}$ with $n_{\max}$ being the maximum number of nodes in a given graph dataset. Further, we use random edge contraction pooling to downsample the graph, and random node-based upsampling to upsample small graphs to larger sizes as described in Section B.1.1. For diversity computations, we choose the shortest-path metric between nodes as a distance function, and the

[2] Available at https://github.com/aidos-lab/magnipy under a BSD 3-Clause Licence.
[3] Available at https://github.com/aidos-lab/mag_edge_pool under a BSD 3-Clause Licence.
[4] The code for computing diversity curves is available at https://github.com/aidos-lab/diversity_curves/.

spread of a graph for computing the diversity. Further, we compute the diversity curves by default over a set of three random coarsening sequences (that is, $R = 3$) , unless specified otherwise. Finally, whenever the number of connected components in a graph is larger than the minimum evaluation scale, we use linear interpolation to interpolate the value of diversity as described in Section 3.1.

B.2 Compute resources

We run our experiments in a compute node with ×2 CPU AMD EPYC 7763 64-CORE PROCESSOR and 1 TiB DRAM. We use these resources to speed up computation, however running experiments on a personal computer is also possible and does not represent a computational bottleneck.

B.3 Baseline methods

For computing the graph embeddings NetLSD [90], FEATHER [74], and Graph2Vec [62] we use the 'Karate Club' Python package [75] published under the MIT licence (GPLv3). The SpectralZoo embeddings we compute via the approximation used by Jin and Zafarani [36][5]. For the computation of the kernel methods we use the Python package GraKel (available under a BSD 3-clause Licence) [79]. For the computation of the TopER[6] method we use the implementation that has been made available with the publication [86]. We use the same approach for the persistent homology based graph embeddings as described in the classification experiment by Ballester and Rieck [2] and use their provided implementation.[7]

C Proofs and theoretical results

Here, we give an overview of all relevant theorems and proofs that detail and support our theoretical results. Specifically, we detail the properties of diversity curves in terms of their computational complexity and their expressivity at distinguishing graphs.

C.1 Computational complexity

Extending our analysis of the time complexity of diversity curve computations in Section 3.2, we outline the computational complexity of the specific steps involved in computing our methods for one edge contraction sequence. Note that the cardinality of the multiset of coarsening sequences over which we compute the diversity curves is constant in practice. Hence, we do not take the averaging over repeated evaluations into account for reporting the total time complexity.

Computing the diversity measure Let C_{Div} be the time complexity for computing the diversity measure of a graph $G = (X, E)$, we conduct our analysis with the spread as it was used in this work. The complexity C_{Div} consists of the time complexity C_{dist} for computing the pairwise distances between all vertices in the graph, and computing the spread. We use the shortest-path distance, which has time complexity $\mathcal{O}(|X|^3)$. In practice, shortest-path distances can be more efficiently computed than alternative graph distances, such as diffusion distances defined in Section B.1.2 and previously used by Limbeck et al. [54]. Further, computing the spread has time complexity $\mathcal{O}(|X|^2)$. This is notably faster than computing alternative diversity measures, such as the magnitude of a graph, which has a time complexity $\mathcal{O}(|X|^3)$. Overall, for a general distance we get $C_{Div} = C_{dist} + \mathcal{O}(|X|^2)$, which in the case of shortest-path distance gives $C_{Div} = \mathcal{O}(|X|^3)$ using Floyd-Warshall algorithm. For our work, we choose to use exact computations of both metric space spread and shortest-path distances as described above to ensure computational accuracy. Nevertheless, we note that costs could further be reduced by approximating the shortest-path distance in around $C_{dist} = O(|X|^{2.032})$ time using a 2-approximation of all shortest-paths [4, 18]. Further, the costs of spread could be reduced via iterative normalisation or mini-batching using s subsets of size $|X_s|$ reducing the time complexity of approximating spread to $C_{Div} = C_{dist} + O(s \cdot |X_s| \cdot |X|)$ [19, 54].

Coarsening the graph Given a graph $G = (X, E)$, let C_{Coarse} be the time complexity for computing the sequence of coarsened graphs $(G_i)_{i \in \mathcal{I}}$ for a fixed set $\mathcal{I}$ of integers representing the

[5] Made available at https://github.com/shengminjin/EstimateSpectralMoments with the publication of their paper [36].
[6] Available at https://github.com/AstritTola/TopER under the MIT Licence
[7] Available at https://github.com/aidos-lab/PH_expressivity under a BSD 3-clause Licence.

cardinalities, so $|G_i| = i$ for any $i \in \mathcal{I}$. We denote by $n_{max} = \max_{i \in \mathcal{I}} i$ the maximum cardinality at which we compute the diversity and $n_{min} = \min_{i \in \mathcal{I}} i$ the minimum cardinality. We obtain the coarsened graphs by doing repeated edge contractions in G and upsampling nodes. The time complexity C_{Coarse} is composed of the time for computing the edge scores, sorting the edges according to their scores, and contracting the edges.

Computing the edge scores If we choose random edge scores, we need $\mathcal{O}(|E|)$ to get all of the edge scores. When using spread-based scores for determining the order of the edge contractions, it takes $\mathcal{O}(|E| \cdot C_{Div})$ time to compute all scores once [54].

Sorting the edges Sorting the edges according to their scores takes $\mathcal{O}(|E| \log(|E|))$ time.

Contracting the edges Contracting one edge has time complexity $\mathcal{O}(|X|)$ and we do this at most $\min(|E|, |X| - n_{min})$ times.

Repeating the edge scoring We might have to repeat the edge score computation and sorting of the edges since we restrict the edge contractions such that no two adjacent edges get contracted in the same iteration of the process of computing edge scores, sorting the edges, and contracting. Let r denote the number of times we have to recompute the edge scores and sort them. This number depends on n_{min} and how many edges can be contracted per iteration. If we want to coarsen the graph such that every edge is contracted, the number of iterations is $r = \log(n / \min(|E|, |X| - n_{min}))$ in the best case for which we can halve the size of the graph in each iteration, and $r = \min(|E|, |X| - n_{min})$ in the worst case where we can contract only one edge per iteration.

Upsampling a node We might also have to upsample graphs. Upsampling one node in a graph of cardinality n has time complexity $\mathcal{O}(n)$. We have to upsample $u = \max(0, n_{max} - |X|)$ nodes for a graph size of at most n_{max}.

Altogether, considering the steps detailed above we arrive at the following costs: For spread-based edge scores, we get a time complexity of $C_{Coarse} = \mathcal{O}(r(|E| \cdot C_{Div} + |E| \log(|E|)) + |E||X| + u \cdot n_{max}) = \mathcal{O}(r|E|(C_{Div} + \log(|E|)) + |E||X| + n_{max}^2)$. This is more expensive than random edge selection. In fact, when generating and sorting random edge scores, we obtain the time complexity $C_{Coarse} = \mathcal{O}(r(|E| + |E| \log(|E|)) + |E||X| + u \cdot n_{max}) = \mathcal{O}(|E|(r \cdot \log(|E|) + |X|) + n_{max}^2)$.

Note that we can further speed up the coarsening process for random edge scores by not first generating random edge scores and sorting them, but rather directly randomly permuting the edges. Then we can continue as above with the ordering of the edges given by the random permutation. This will improve the time complexity for computing the sequence of coarsened graphs using random edge scores to $C_{Coarse} = \mathcal{O}(|E|(r + |X|) + u \cdot n_{max}) = \mathcal{O}(|E|(r + |X|) + n_{max}^2)$. Note that in case $n_{max} < |X|$, we do not need to do any upsampling, and the time complexity for random edge contractions further reduces to $C_{Coarse} = \mathcal{O}(|E|(r + |X|))$.

Computing the diversity curves To compute diversity curves, we calculate the diversity measure at every cardinality $i \in \mathcal{I}$ for graphs with at most n_{max} nodes, which takes $\mathcal{O}(|\mathcal{I}| \cdot n_{max}^3)$ time when using spread. In total, the time complexity of computing the diversity curves is given by

- $C_{Coarse} + C_{Div} = \mathcal{O}(r|E|(C_{Div} + \log(|E|)) + |E||X| + |\mathcal{I}| \cdot n_{max}^3)$ using spread-based edge scores,
- $C_{Coarse} + C_{Div} = \mathcal{O}(|E|(r \cdot \log(|E|) + |X|) + |\mathcal{I}| \cdot n_{max}^3)$ when sorting random edge scores, or
- $C_{Coarse} + C_{Div} = \mathcal{O}(|E|(r + |X|) + |\mathcal{I}| \cdot n_{max}^3)$ when permuting edges randomly.

Overall, we find that by using coarsening as one of their core components, diversity curves remain reasonably efficient to compute in practice. Further, our framework allows for investigating even faster coarsening methods or graph invariants in future work.

C.2 Diversity curves collapse to the same diversity

In this subsection, we describe how diversity curves behave if we coarsen graphs as much as possible by performing consecutive edge contractions until there are no more edges to contract. This illustrates that eventually, all diversity curves will reach the same values at low cardinalities. The proof of this theorem uses that our chosen graph coarsening procedure does not change the connectivity of the graph.

Theorem 2. *Any two graphs G and H with the same number c of connected components will collapse to the same graph of cardinality c and their diversity curves agree for $i \in \{1, \dots c\}$. Concretely,* $\mathrm{DivCurve}(G)_i = \mathrm{DivCurve}(H)_i = i$. *Furthermore,* $\mathrm{DivCurve}(H)_{c+1} = \mathrm{DivCurve}(G)_{c+1} \neq c + 1$.

Proof. Consider a graph G with exactly c connected components. Doing edge contractions in a graph does not change its number of connected components. For any possible sequence of edge contractions, we will arrive at the graph $\overline{K_c}$, the graph on c vertices with no edges. The spread of this graph is $\mathrm{Div}(\overline{K_c}) = c$. In particular, a connected graph will collapse to $\overline{K_1}$, the graph on 1 vertex, with spread $\mathrm{Div}(\overline{K_1}) = 1$. For graphs with more than one connected component, we perform the linear interpolation, which gives us values of $\mathrm{DivCurve}(\overline{K_c})_i = i$ for $1 \leq i < c$. For any two graphs G and H with the same number c of connected components we thus have $\mathrm{DivCurve}(G)_i = \mathrm{DivCurve}(H)_i = i$ for $i \in \{1, \dots c\}$. Note also that since the number of connected components is c, for a graph with $c+1$ vertices this is possible only if there exists precisely one edge. The spread of this graph is $c - 1 + \frac{2}{1+\exp(-1)}$. Hence, we have shown that $\mathrm{DivCurve}(H)_{c+1} = \mathrm{DivCurve}(G)_{c+1} \neq c+1$. □

C.3 Diversity curves over all possible edge contractions

For the remaining two sections we study diversity curves over all possible edge contraction sequences. That is, the multiset over which we average the diversity curves contains all possible coarsenings obtained from edge-contraction sequences as described in the paragraph below. Theoretically, this allows us to study diversity curves without the need for a canonical labelling of the edges, a notoriously difficult problem related to the graph isomorphism problem [1]. In practice, the evaluation of all possible edge contraction sequences becomes infeasible. Hence, we empirically compute diversity curves over a small number of edge contraction sequences. Section D.8 verifies that this empirical approach is nevertheless stable with respect to the chosen edge contraction sequence and the number of evaluated sequences.

For a graph G and a sequence $s_e = (e_i)_{1 \leq i \leq k}$ of $m \geq k \geq 0$ of edges in G, we denote by $G/(e_i)_{1 \leq i \leq k}$ the graph obtained from G after iteratively contracting the edges in the sequence s_e. We say that the sequence of coarsened graphs $\mathcal{G}_\mathcal{I} = (G_i)_{i \in \mathcal{I}}$ is induced by the edge sequence $(e_i)_{i \leq k}$ if $\mathcal{I} = \{n-k, \dots, n\}$ and $G_i = G/(e_j)_{1 \leq j \leq n-i}$ for any $i \in \mathcal{I}, i < n$ and $G_n = G$. We consider the multiset of all valid edge contraction sequences in the graph G of length $0 \leq k \leq m$ by $\mathcal{E}_k(G) = \{\{ (e_i)_{i \leq k} | \forall i \leq k \; e_i \in E(G/(e_j)_{j \leq i-1}) \}\}$. In a previous version of the manuscript, we considered the multiset of diversity curves associated to each sequence of k edge contractions in order to study the theoretical properties, that is $\{\{\mathrm{DivCurve}(\mathcal{G}_\mathcal{I}) | \exists (e_i)_{i \in \mathcal{I}} \in \mathcal{E}_k$ such that $\mathcal{G}_\mathcal{I}$ is induced by $(e_i)_{i \in \mathcal{I}}\}\}$, denoted by the multisets of diversity curves under all possible edge contractions. From this formulation we obtain the diversity curve over all possible edge contractions under the current definition by taking for each cardinality the average with multiplicities over all values of the diversity curves appearing in the multiset in that cardinality. The current formulation allows us to treat diversity curves in a more unified way.

C.4 Diversity curves as graph invariants

Theorem 1. *For graphs $G \cong H$, their diversity curves over all possible edge contractions are equal.*

Proof. The statement follows immediately from the fact that the spread of a graph is invariant under graph isomorphism, and that for two isomorphic graphs $G \cong H$ where we denote by $\psi : G \to H$ an isomorphism, contracting an edge e in G and the corresponding edge $\psi(e)$ in H gives two isomorphic graphs. Hence for any edge sequence $(e_i)_{i \leq k}$ of G and the corresponding coarsening sequence of G, the edge sequence $(\psi(e_i))_{i \leq k}$ defines a sequence of coarsened graphs that are isomorphic to the coarsened graphs of G for every $i \leq k$. Hence the diversity at each cardinality is equal and because of the isomorphism we obtain a bijection between the two multisets of possible edge contractions, which implies the average diversity at each cardinality is equal. □

Even when not considering the diversity curves over all possible edge contractions, we can state the following:

Proposition 1. *Let G and H be two isomorphic graphs and σ a multiset of edge sequences in G and the induced multiset of coarsened graphs $\mathcal{G}$. There exists a multiset of edge sequences in H with induced multiset of coarsened graphs $\mathcal{H}$ such that* $\mathrm{DivCurve}(\mathcal{G}) = \mathrm{DivCurve}(\mathcal{H})$.

Proof. Let $\psi : G \to H$ be an isomorphism between the two graphs. The multiset $\sigma_H = \{\{(\psi(e_i))_{1 \leq i \leq k} | (e_i)_{1 \leq i \leq k} \in \sigma\}\}$ of edge sequences in H ensures that for every edge sequence

in σ there is exactly one edge sequence in σ_H where the coarsened graphs at each cardinality are equal and vice versa. Hence the diversity curve of H over σ_H equals the diversity curve of G over σ. □

Hence, even if we can not compute all possible edge sequences, there always exists a corresponding multiset of edge sequences such that the diversity curves are equal, and if we choose the edges in a way that is respecting an isomorphism, we will find the corresponding sequence.

C.5 Diversity curves increase expressivity

In this section, we want to study the expressivity of the diversity curves and spread, combining properties of edge contraction coarsening and the graph invariant itself. Note that it is known that several classes of graphs can be reconstructed based on their multiset of single edge contractions [71], while the spread of graphs has empirically demonstrated the potential for being an expressive graph invariant [54]. Altogether, as detailed in Section 3.2 and Section C.4, we consider the diversity curves over all possible edge contractions for this discussion. Further, we define what it means to increase expressivity in Definition 1 following existing work by Lachi et al. [46], who previously showed that edge contraction pooling layers do not only preserve, but also increase expressivity. We do restrict to consider graphs of the same cardinality. Our first result gives an example which shows that tracking the spread over a coarsening of graphs increases the expressivity compared to just considering the spread at the original cardinality of the graphs.

Theorem 3. *The diversity curves over all possible edge contractions is more expressive than spread.*

Proof. It is clear that diversity curves are at least as expressive as spread, meaning that for any two graphs G and H of the same cardinality with $\mathrm{Div}(G) \neq \mathrm{Div}(H)$, also $\mathrm{DivCurve}(G) \neq \mathrm{DivCurve}(H)$ since $\mathrm{DivCurve}(G)_{|G|} = \mathrm{Div}(G) \neq \mathrm{Div}(H) = \mathrm{DivCurve}(H)_{|H|}$ for any coarsening of G and H respectively, which implies that the diversity curves over all possible edge contractions differ, since every diversity curve over a coarsening of G differs in at least one component from a diversity curve over any coarsening of H.
Consider the two graphs G_A, G_B in Figure S.1, where G_A is the graph on the left, and G_B is the so-called house graph on the right. They have equal spread $\mathrm{Div}(G_A) = \mathrm{Div}(G_B) = \frac{2}{1+3\exp(-1)+\exp(-2)} + \frac{3}{1+2\exp(-1)+2\exp(-2)} \approx 2.39$. The one-edge contractions of G_A are all isomorphic to the diamond graph D, the four-cycle with one diagonal edge, which has spread $\mathrm{Div}(D) = \frac{2}{1+2\exp(-1)+\exp(-2)} + \frac{2}{1+3\exp(-1)} \approx 2.02$, which consequently is the diversity curve of G_A over all possible edge contractions at cardinality 4. The one edge contractions of G_B consist of the four-cycle C_4 with multiplicity 2 and spread $\frac{4}{1+2\exp(-1)+\exp(-2)} \approx 2.14$, the diamond graph D with multiplicity 3 and the three-cycle with an additional edge to a new vertex with multiplicity 1 and spread $\frac{1}{1+\exp(-1)+2\exp(-2)} + \frac{2}{1+2\exp(-1)+\exp(-2)} + \frac{1}{1+3\exp(-1)} \approx 2.15$, giving us an average in diversity over all one-edge contractions of $\approx 2.08 > \mathrm{Div}(D)$, showing that the diversity curves over all possible edge contractions differ already at cardinality 4. The diagram on the right of Figure S.1 shows the different diversity curves over all possible edge contractions at each cardinality for the two graphs G_A and G_B. □

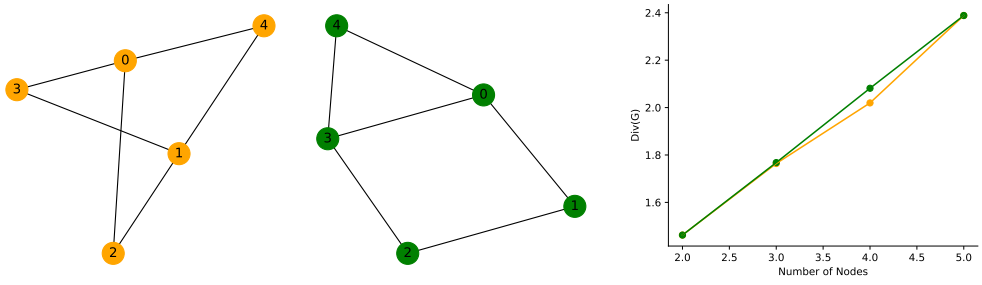

Figure S.1: Graphs with identical diversity, $\mathrm{Div}(G)$ as computed via spread using shortest-path distances, that can be distinguished through edge pooling by their diversity curves, $\mathrm{DivCurve}(G)$. This example is used in Theorem 3 and Proposition 2.

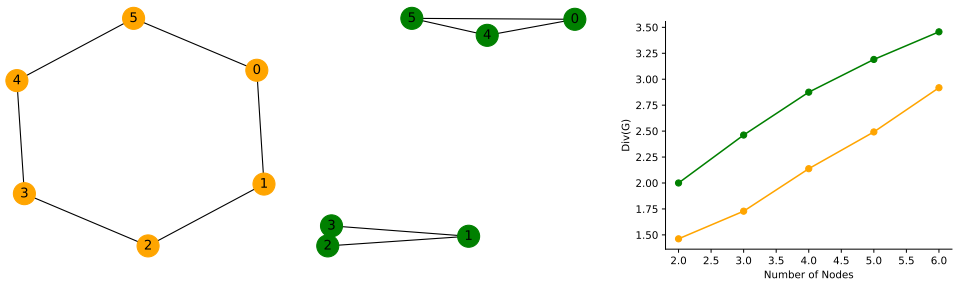

Figure S.2: Graphs that are 1-WL indistinguishable, but can be distinguished through their diversity, $\mathrm{Div}(G)$, or by their diversity curves, $\mathrm{DivCurve}(G)$. This example is used in Proposition 2.

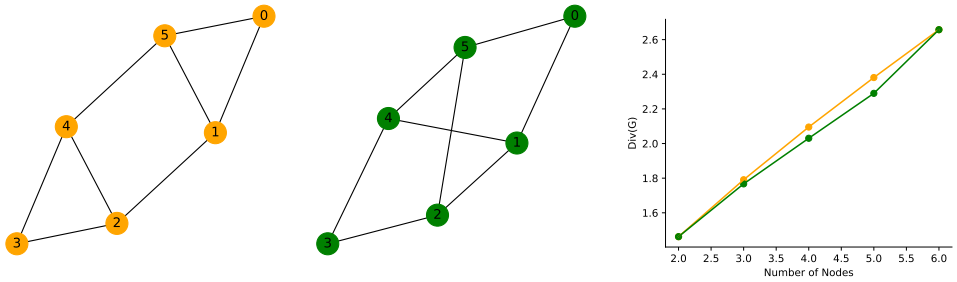

Figure S.3: Graphs that are 1-WL indistinguishable and have identical diversity, $\mathrm{Div}(G)$ as computed via spread using shortest-path distances, that can be distinguished through edge pooling by their diversity curves, $\mathrm{DivCurve}(G)$. This example is used in Proposition 3.

For our next statement, we first introduce another notion of comparing expressivity of two functions.

Definition 7. *For two functions φ and ψ on the set of graphs, we say that φ and ψ are* incomparable *if there exist graphs G_A, G_B, G_C, and G_D such that $\varphi(G_A) \neq \varphi(G_B)$ and $\psi(G_A) = \psi(G_B)$, and $\varphi(G_C) = \varphi(G_D)$ and $\psi(G_C) \neq \psi(G_D)$.*

Proposition 2. *The spread and the Weisfeiler-Leman (1-WL) test are incomparable in terms of expressivity.*

Proof. We give concrete examples of two pairs of graphs. First, consider the disjoint union of two three-cycles $C_3 \sqcup C_3$ and the six-cycle C_6, which are known to be indistinguishable by 1-WL. The spread of the disjoint union of two three-cycles is $\mathrm{Div}(C_3 \sqcup C_3) = \frac{6}{1+2\exp(-1)} \approx 3.46$, whereas the spread of the 6-cycle is $\mathrm{Div}(C_6) = \frac{6}{1+2\exp(-1)+2\exp(-2)+\exp(-3)} \approx 2.92$. Figure S.2 illustrates this example.
Second, consider again the two graphs in Figure S.1, which have the same spread as we have seen in the proof of Theorem 3. Those two graphs are distinguishable by 1-WL, since the degrees of the neighbourhoods of the vertices of degree 3 differ in the two graphs. Hence, we show that the spread and 1-WL are incomparable in terms of expressivity. □

Proposition 3. *There exists a pair of graphs that is 1-WL-indistinguishable, indistinguishable by their spread alone, but distinguishable by their diversity curves over all possible edge contractions.*

Proof. Consider the two graphs G_C and G_D in Figure S.3, where G_C is the 6-cycle graph with the additional edges $(1,5)$ and $(2,4)$ on the left, and G_D is the 6-cycle graph with the additional edges $(1,4)$ and $(2,5)$ on the right. They are indistinguishable by the 1-WL test as the colouring obtained from the degrees of each node is already a stable colouring. Both graphs have the same spread, concretely $\mathrm{Div}(G_C) = \mathrm{Div}(G_D) = \frac{2}{1+2\exp(-1)+2\exp(-2)+\exp(-3)} + \frac{4}{1+3\exp(-1)+2\exp(-2)} \approx 2.66$. As is shown in Figure S.3 on the right, the average spread over all possible edge contractions differs already after one edge contraction in each graph. □

In total, these results show that if we have a pipeline that already distinguishes graphs on the level of 1-WL, then adding the diversity curves will increase the expressivity. Empirically, we demonstrate this in Section D.6 and Section 4.6.

D Extended experiments

In the following section, we describe extended experimental details of all of our experiments, which describe how to reproduce our results. Further, we provide empirical expressivity results and ablation studies that support our main results.

D.1 Graph perturbations

The perturbation experiment in Section 4.1 applies one of the following perturbation scenarios: (i.) ADD_EDGE (▲) addition of an edge, (ii.) REMOVE_EDGE (■) removal of a non-bridge edge (iii.) REWIRE_EDGE (●) connect one of the endpoints of an edge to another vertex and (iv.) SWAP_EDGE (◆) given two edges (v_0, v_1) and (v_2, v_3) construct two new edges as (v_0, v_2) and (v_1, v_3). The perturbation degree refers to the percentage of edges that were perturbed. In each case this represents a different quantity. In (i.) the degree p is the percentage of edges remaining before the graph is fully-connected, i.e. $p = 1$ means enough edges have been added so that the graph is fully-connected. In (ii.) p is the percentage of edges that are not bridges, i.e. edges that do not result in the creation of more connected components when removed. In cases (iii.) and (iv.), p is the percentage of edges in each graph that are altered. We uniformly sample edges from the pool of available ones. Across each perturbation scenario and perturbation degree $p \in \{0.1, 0.2, \ldots, 1\}$, we apply the perturbation and track the norm of the average diversity curve across graphs, computed using shortest-path distances at node numbers between one and the maximum number of nodes in each dataset. Specifically, we consider the following datasets: `PLANAR` consists of 200 graphs with 64 nodes each, `LOBSTER` consists of 128 graphs with parameters $p_1 = p_2 = 0.7$ with a minimum number of 10 nodes and a maximum number of 100 nodes, `SBM` consists of 200 graphs with 20-40 nodes per community with 2-5 communities subset to all graphs with less than 60 nodes, and `COMM20` consists of 200 graphs of a stochastic block model with 2 communities [58][8]. The additional dataset `EGO` consists of 757 graphs extracted from Citeseer [35] [9]. We take a maximum of 80 graphs from each

[8] We take the code for generating the datasets from https://github.com/KarolisMart/SPECTRE available under an MIT licence.

[9] We take the code for generating these datasets from https://github.com/yunhuijang/GEEL

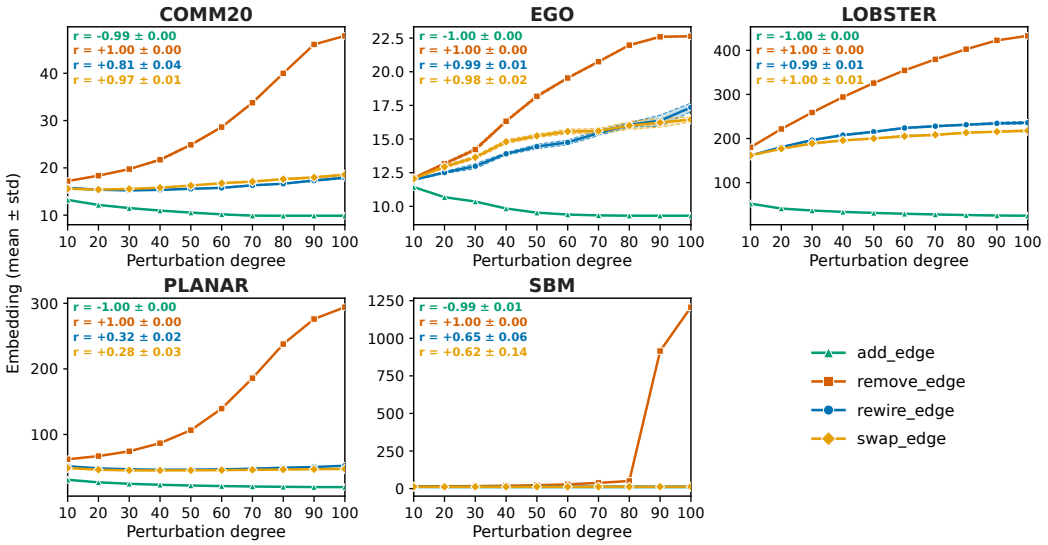

Figure S.4: Diversity curves track the change in graph structure across edge perturbations. We report the norm of the mean diversity curve per dataset by perturbation degree.

dataset, which is the test split except for the case of SBM, for which we consider 40 graphs with less than 60 nodes. Figure S.4 and Figure 2 show the results averaged over 5 random seeds along with the standard deviation.

D.2 Simulated graph distributions

Following our motivation of characterising graphs drawn from the same underlying distribution but across varying cardinalities, we elaborate on our results from Section 4.2.

D.2.1 Distinguishing graph distributions

In Section 4.2, we investigate how well diversity curves can distinguish between graphs generated from different random graph distributions. Specifically, we generate graphs with different sizes using the Python package NetworkX [28] released under a BSD-3-Clause Licence. We choose the following four random graph distributions, for which we each generate 3 graphs for every cardinality $n \in \{10, \dots, 29\}$:

- Erdős-Rényi (ER) graphs $G_{n,p}$ with edge probability parameter $p = 0.75$.
- Random partition (RP) graphs on n vertices, which have communities of fixed sizes and edge probabilities p_{in} for nodes in the same community and p_{out} for nodes in different communities. We fix the community sizes of the RP graphs such that we have 3 communities, each of about $1/3$ the size of the graph and we set $p_{in} = 0.9$, and $p_{out} = 0.1$.
- Random geometric (RG) graphs, which are generated by choosing n nodes uniformly at random in the unit square and adding an edge for two nodes if their Euclidean distance is less than or equal to a parameter r, which we set to $r = 0.25$.
- Stochastic block model (SBM) of n vertices, which partitions the nodes into communities and adds edges according to the community the vertices belong to. As in the RP model, we choose probabilities $p_{in} = 0.8$ and $p_{out} = 0.05$ which specify the edge probabilities between vertices in the same cluster and different clusters respectively. For the communities we partition the node set into 4 communities each of about $1/4$ of the graph's size.

Example graphs generated by each distribution are shown in Figure S.5. Based on the graphs generated as detailed above, we aim to assess the ability of our method to distinguish between graph distributions. For computing the classification accuracies, we use a kNN-classifier with fixed $k = 5$ and 10-fold stratified grouped cross validation, where we group the graphs of the same cardinalities together. To assess the clustering qualities, we report the silhouette score, Calinski–Harabasz index, and Davies–Bouldin index with the true class labels, as well as the adjusted rand indices (ARI) based on k-means clustering and a hierarchical clustering approach. For the k-means clustering we specify the correct number of clusters and use 'k-means++' for the initialisation method. For the hierarchical clustering we use agglomerative clustering where we specify the correct number of clusters and as a linkage criterion we use the average over all distances between the points in two clusters. All parameters which we do not specify here are used with their default setting as implemented in scikit-learn (version 1.5.2). To obtain uncertainties for the clustering metrics, we take the mean and standard deviation over 100 random resamples using 50% of the data points each.

Diversity curve computations. The kNN-classifier as well as the distances with which we compute clustering quality scores use the L^2-norm between the diversity curves viewed as vectorial embedding

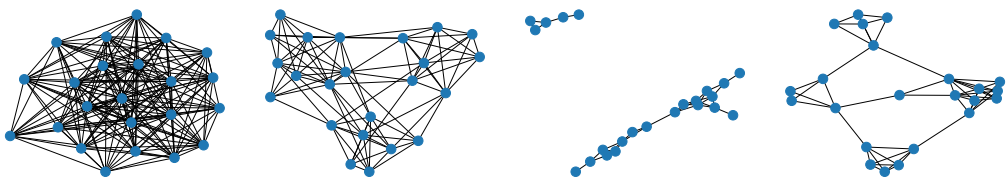

Figure S.5: Graphs drawn from the distributions ER, RP, RG, SBM used in the experiment to distinguish different distributions from each other.

Table S.1: Accuracies and clustering scores for distinguishing between the graph distributions ER, RP, SBM, and RG.

	Method	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI k-means	ARI agglomerative
Diversity Curves	Shortest Path	**0.904 ± 0.059**	0.423 ± 0.034	0.753 ± 0.060	191.237 ± 33.144	0.495 ± 0.071	0.313 ± 0.111
	Shortest Path PCA	0.892 ± 0.065	**0.432 ± 0.035**	**0.737 ± 0.060**	**194.064 ± 34.006**	**0.500 ± 0.075**	0.323 ± 0.123
Diversity Function	Spread Function Shortest Path	0.829 ± 0.029	0.033 ± 0.019	3.949 ± 0.390	12.517 ± 2.420	0.097 ± 0.050	0.095 ± 0.049
Embeddings	NetLSD	0.900 ± 0.033	0.055 ± 0.038	1.332 ± 0.112	38.799 ± 7.908	0.150 ± 0.054	0.092 ± 0.045
	FEATHER	0.750 ± 0.091	0.108 ± 0.023	1.914 ± 0.287	74.719 ± 8.443	0.293 ± 0.033	0.277 ± 0.038
	SpectralZoo	0.721 ± 0.088	0.050 ± 0.031	2.875 ± 2.025	33.794 ± 4.979	0.257 ± 0.048	0.140 ± 0.082
	Graph2Vec	0.504 ± 0.094	-0.005 ± 0.001	7.365 ± 0.291	1.405 ± 0.102	0.023 ± 0.024	0.002 ± 0.008
Kernel	Kernel WL Optimal Assignment	0.900 ± 0.046	0.019 ± 0.004	-	-	-	0.259 ± 0.046
	Kernel Core Framework	0.842 ± 0.064	0.249 ± 0.026	-	-	-	0.446 ± 0.077
	Kernel Core Framework MDS	0.821 ± 0.038	0.266 ± 0.026	1.414 ± 0.178	91.392 ± 11.984	0.459 ± 0.049	**0.472 ± 0.060**
	Kernel WL Optimal Assignment MDS	0.796 ± 0.101	0.045 ± 0.020	3.491 ± 0.551	13.624 ± 2.199	0.185 ± 0.045	0.202 ± 0.057
	Kernel WL MDS	0.779 ± 0.072	0.239 ± 0.034	2.531 ± 0.443	21.090 ± 4.602	0.429 ± 0.061	0.273 ± 0.094
	Kernel Shortest Path	0.771 ± 0.073	0.284 ± 0.044	-	-	-	0.426 ± 0.057
	Kernel Shortest Path MDS	0.767 ± 0.046	0.274 ± 0.038	1.597 ± 0.169	57.105 ± 11.254	0.444 ± 0.065	0.423 ± 0.064
	Kernel WL	0.746 ± 0.094	0.213 ± 0.029	-	-	-	0.031 ± 0.090
	Kernel Pyramid Match	0.671 ± 0.039	0.002 ± 0.009	-	-	-	0.017 ± 0.016
	Kernel Pyramid Match MDS	0.588 ± 0.051	-0.009 ± 0.012	7.657 ± 3.929	7.242 ± 1.145	0.065 ± 0.038	0.045 ± 0.032
	Kernel Neighbourhood Hash	0.500 ± 0.062	-0.032 ± 0.006	-	-	-	0.002 ± 0.008
	Kernel Neighbourhood Hash MDS	0.421 ± 0.066	-0.066 ± 0.008	20.290 ± 10.776	0.683 ± 0.411	-0.007 ± 0.008	-0.003 ± 0.007
TopER	TopER Degree	0.758 ± 0.091	-0.064 ± 0.021	5.288 ± 1.902	31.191 ± 6.050	0.089 ± 0.019	0.089 ± 0.024
	TopER Closeness	0.746 ± 0.066	-0.093 ± 0.019	8.445 ± 7.621	25.482 ± 5.115	0.084 ± 0.020	0.076 ± 0.020
	TopER All Filtrations	0.700 ± 0.096	-0.055 ± 0.020	3.255 ± 0.499	27.534 ± 5.504	0.101 ± 0.025	0.073 ± 0.028
	TopER Popularity	0.696 ± 0.093	-0.022 ± 0.037	1.344 ± 0.129	29.667 ± 7.188	0.113 ± 0.076	0.062 ± 0.019
	TopER Forricci	0.646 ± 0.088	-0.067 ± 0.022	2.950 ± 1.701	31.450 ± 5.224	0.112 ± 0.024	0.112 ± 0.027
	TopER Olricci	0.592 ± 0.069	-0.099 ± 0.015	10.200 ± 9.354	13.468 ± 3.562	0.059 ± 0.020	0.043 ± 0.017
Persistent Homology	PH Vietoris Rips	0.783 ± 0.067	-0.010 ± 0.031	2.227 ± 0.175	16.156 ± 2.031	0.085 ± 0.028	0.053 ± 0.029
	PH Laplacian	0.783 ± 0.074	-0.132 ± 0.018	2.668 ± 0.247	11.701 ± 2.184	0.055 ± 0.021	0.034 ± 0.019
	PH Alpha Complex	0.742 ± 0.103	0.014 ± 0.021	1.992 ± 0.182	43.715 ± 6.740	0.202 ± 0.034	0.160 ± 0.054
	PH Fricci	0.742 ± 0.067	-0.099 ± 0.020	2.498 ± 0.220	30.874 ± 4.917	0.090 ± 0.031	0.043 ± 0.029
	PH Degree	0.725 ± 0.062	-0.160 ± 0.019	3.016 ± 0.353	8.580 ± 1.651	0.040 ± 0.020	0.018 ± 0.014
	PH Olricci	0.696 ± 0.085	-0.125 ± 0.023	2.945 ± 0.473	34.161 ± 5.567	0.092 ± 0.031	0.060 ± 0.023

in $\mathbb{R}^{29}$ across evaluation scales from 1 to 29. Results are reported in Table 1. For visualisation as in Figure 3, we perform PCA on the diversity curves using the L^2-norm.

Spread functions as baselines. The spread function, as introduced by Willerton [93], measures how the spread of a metric space changes with respect to a scale factor t of the distance. Concretely, for a graph G viewed as a metric space with shortest-path distance d_G and a parameter $t > 0$, we define a metric space tG with the same underlying points as G and with the scaled metric $t \cdot d_G$. The spread function is the spread $\mathrm{Div}(tG)$ as a function of $t > 0$. As an ablation, we compare our diversity curves to this approach of the diversity function. For each graph, we compute the spread function at the graph's original size for $1 \leq t \leq 5$ on 20 evenly spaced evaluation scales and report the classification accuracies and clustering metrics using the L^2-norm between the spread functions.

Graph embedding baselines. We compare our results to relevant groups of unsupervised graph representation methods, namely graph kernels, spectral graph representations given by NetLSD [90] and SpectralZoo [36], the topological methods TopER [86] and a persistent homology based embedding [2], as well as the graph embeddings FEATHER [74] and Graph2Vec [62]. For all the graph embeddings, NetLSD, FEATHER, and Graph2Vec, we use the vectorial representations with Euclidean distances for computing the clustering accuracies and scores.

Kernel baselines. For the graph kernels, we compute the Kernel matrix K and use $-K + \max_{i,j} K_{i,j}$ as a pairwise distance to obtain clustering accuracies, silhouette scores, and the adjusted rand index with agglomerative clustering. We also use these pairwise distances for dimensionality reduction with MDS and compute all of our clustering metrics and accuracies also on the transformed data.

Topological baselines. For TopER we report for each of the filtrations described by Tola et al. [86] as sublevel filtrations individually and also combine all of them. We use the vectorial representations with Euclidean distances to compute clustering accuracies and scores. We use the same approach for the persistent homology based graph embeddings as described in the classification experiment by Ballester and Rieck [2]. Namely, we use different filtrations to compute persistence homology up to dimension two on the clique complexed obtained from the graphs by filling in every 3-clique, so we obtain a simplicial 2-complex. We transform the resulting persistence diagrams into a vectorial representation using their persistent images. We again use the Euclidean distances of the embeddings to compute clustering accuracies and scores.

Table S.2: Accuracies and clustering scores of baseline structural summaries for distinguishing between the graph distributions ER, RP, SBM, and RG.

	Method	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI k-means	ARI agglomerative
Diversity Curves	Shortest Path	**0.904 ± 0.059**	0.423 ± 0.034	0.753 ± 0.060	191.237 ± 33.144	0.495 ± 0.071	0.313 ± 0.111
	Shortest Path PCA	0.892 ± 0.065	**0.432 ± 0.035**	**0.737 ± 0.060**	**194.064 ± 34.006**	**0.500 ± 0.075**	**0.323 ± 0.123**
Baseline	Summary Statistic	0.838 ± 0.029	-0.053 ± 0.017	3.196 ± 0.640	31.287 ± 5.296	0.111 ± 0.024	0.090 ± 0.033
	Laplacian Spectrum (binned)	0.838 ± 0.060	0.058 ± 0.011	2.785 ± 0.237	20.417 ± 2.458	0.215 ± 0.051	0.150 ± 0.066
	Degree Sequence	0.833 ± 0.046	-0.028 ± 0.012	3.459 ± 0.308	23.817 ± 3.310	0.135 ± 0.044	0.091 ± 0.034
	Average Shortest Path	0.762 ± 0.059	0.187 ± 0.040	23.435 ± 90.534	30.409 ± 7.871	0.378 ± 0.044	0.305 ± 0.050
	Clustering Coefficient	0.617 ± 0.095	0.041 ± 0.031	4.252 ± 4.872	34.702 ± 5.473	0.235 ± 0.055	0.129 ± 0.091
	Max Diameter	0.579 ± 0.101	0.243 ± 0.043	15.235 ± 31.362	33.867 ± 6.923	0.378 ± 0.091	0.169 ± 0.072
	Number of Edges	0.483 ± 0.121	-0.094 ± 0.021	3.666 ± 2.338	33.508 ± 5.866	0.113 ± 0.024	0.089 ± 0.029
	Number of Connected Components	0.454 ± 0.060	-0.134 ± 0.055	1.713 ± 0.409	68.197 ± 14.380	0.168 ± 0.032	0.186 ± 0.032
	Number of Nodes	0.250 ± 0.000	-0.083 ± 0.013	99.298 ± 196.613	0.532 ± 0.474	-0.012 ± 0.006	-0.012 ± 0.007

The complete results can be found in Table S.1, which is an extended version of Table 1. Diversity curves perform similar in accuracy as the best performing baselines, and outperform the baselines in terms of almost all reported clustering metrics.

Comparison to simple structural baselines As a sanity check, we compare the performance of diversity curves in the simulation experiment from Section 4.2 as reported in Table S.2 against simple structural baselines. Concretely, we consider the number of nodes, number of edges, number of connected components, average clustering coefficient, the maximum diameter over all connected components, the average shortest path distance (averaged over all connected components), the degree statistic (degree sequence padded with zeros to obtain the same length for all graphs), and a vectorisation of the histogram of the normalised Laplacian spectrum, and use each of these inputs to perform the kNN-classification and compute the clustering scores. Additionally, we also report the accuracy and clustering performance when collecting all the structural invariants in a summary statistic. As expected, we see that the classification based on the number of nodes is based on chance, confirming that by construction the number of nodes is uninformative for the tasks at hand.

We further observe that all tested structural baselines perform worse in terms of the clustering metrics compared to diversity curves. Also in terms of accuracies, diversity curves outperform all structural baselines, with the normalised Laplacian spectrum achieving the highest accuracy amongst the structural baselines after the summary statistic. Hence, we verify that diversity curves provide information beyond standard graph summary statistics for these tasks.

D.2.2 Distinguishing parameter choices

Next, we evaluate how well our methods can distinguish between different parameters used to generate graphs. We look at one random graph model at a time and vary the parameter choices within the model. We again generate 3 graphs per class for every node size $n \in \{10, \dots, 29\}$ to understand how well the diversity curves can distinguish between these more subtle changes in the distribution while being robust to variations of the sizes of the graphs. We use the same graph models as in Section D.2 and vary the parameters as follows:

- Erdős-Rényi (ER) $G_{n,p}$ where we vary the edge probability $p \in \{0.1, 0.5, 0.9\}$;
- Random partition (RP) graphs where we vary the edge probabilities $(p_{in}, p_{out}) \in \{(0.9, 0.01), (0.9, 0.1), (0.9, 0.5)\}$ and fix the number of communities to be 3;
- Random geometric (RG) graphs where we vary the radius $r \in \{0.25, 0.5, 0.75\}$ determining which pairs of nodes are connected by an edge;
- Stochastic block model (SBM), where we vary the number of communities between 2, 3, and 4 roughly equally sized communities. The edge probabilities $p_{in} = 0.6$ and $p_{out} = 0.01$ are fixed.

For each random graph model, we compute the classification accuracies on the test set and clustering scores as described in Section D.2 for the same set of baseline methods. The full results can be found in Tables S.3, S.4, S.5, and S.6. Furthermore, we show the PCA plot of the diversity curves for each distribution in Figure S.6, which shows that diversity curves are able to cluster the graphs belonging to the same parameter setting clearly.

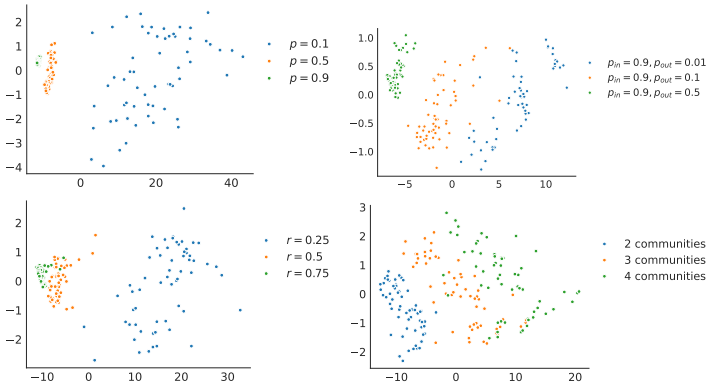

Figure S.6: PCA plots of the diversity curves for ER, RP, RG, and SBM graphs.

Table S.3: Accuracies and clustering scores for distinguishing parameter choices in ER graphs

	Method	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI agglomerative	ARI k-means
Diversity Curves	Shortest Path	**1.000 ± 0.000**	0.679 ± 0.021	0.346 ± 0.018	272.892 ± 54.881	0.486 ± 0.053	0.444 ± 0.056
	Shortest Path PCA	**1.000 ± 0.000**	0.693 ± 0.021	**0.328 ± 0.019**	277.053 ± 56.295	0.485 ± 0.052	0.445 ± 0.056
Diversity Function	Spread Function Shortest Path	0.956 ± 0.069	0.133 ± 0.019	2.209 ± 0.327	29.456 ± 5.114	0.198 ± 0.097	0.189 ± 0.067
Embeddings	FEATHER	**1.000 ± 0.000**	0.574 ± 0.024	0.698 ± 0.053	186.237 ± 23.639	0.811 ± 0.134	0.800 ± 0.074
	SpectralZoo	**1.000 ± 0.000**	0.335 ± 0.027	1.039 ± 0.107	161.364 ± 26.514	0.453 ± 0.115	0.439 ± 0.023
	NetLSD	0.972 ± 0.051	0.096 ± 0.027	2.044 ± 0.196	45.743 ± 8.983	0.076 ± 0.062	0.194 ± 0.062
	Graph2Vec	0.589 ± 0.083	-0.001 ± 0.002	7.240 ± 0.335	1.328 ± 0.107	0.001 ± 0.005	0.011 ± 0.024
Kernel	Kernel Shortest Path	**1.000 ± 0.000**	**0.793 ± 0.035**	-	-	0.857 ± 0.066	-
	Kernel Core Framework	**1.000 ± 0.000**	0.644 ± 0.019	-	-	0.778 ± 0.235	-
	Kernel Shortest Path MDS	0.994 ± 0.017	0.696 ± 0.034	0.482 ± 0.072	180.703 ± 47.049	0.758 ± 0.114	0.781 ± 0.064
	Kernel Core Framework MDS	0.994 ± 0.017	0.670 ± 0.018	0.445 ± 0.026	**356.006 ± 44.846**	**0.940 ± 0.093**	**0.942 ± 0.100**
	Kernel WL MDS	0.994 ± 0.017	0.557 ± 0.032	0.678 ± 0.098	97.450 ± 22.580	0.477 ± 0.044	0.765 ± 0.144
	Kernel WL	0.989 ± 0.022	0.513 ± 0.030	-	-	0.459 ± 0.060	-
	Kernel WL Optimal Assignment	0.978 ± 0.037	0.091 ± 0.009	-	-	0.586 ± 0.080	-
	Kernel WL Optimal Assignment MDS	0.917 ± 0.067	0.241 ± 0.030	1.973 ± 0.410	29.540 ± 4.918	0.478 ± 0.122	0.414 ± 0.136
	Kernel Pyramid Match	0.811 ± 0.132	0.035 ± 0.011	-	-	0.017 ± 0.028	-
	Kernel Pyramid Match MDS	0.756 ± 0.106	0.056 ± 0.015	11.737 ± 9.093	6.665 ± 1.152	0.050 ± 0.031	0.036 ± 0.045
	Kernel Neighbourhood Hash	0.611 ± 0.122	-0.007 ± 0.007	-	-	0.006 ± 0.014	-
	Kernel Neighbourhood Hash MDS	0.556 ± 0.082	-0.032 ± 0.010	14.111 ± 14.751	2.214 ± 1.022	0.012 ± 0.020	0.019 ± 0.025
TopER	TopER Forricci	0.906 ± 0.066	0.263 ± 0.044	1.196 ± 0.185	56.032 ± 10.377	0.180 ± 0.097	0.292 ± 0.077
	TopER Closeness	0.894 ± 0.058	0.057 ± 0.037	2.578 ± 0.876	20.609 ± 4.163	0.101 ± 0.053	0.133 ± 0.047
	TopER All Filtrations	0.894 ± 0.084	0.139 ± 0.037	2.062 ± 0.474	30.453 ± 5.249	0.145 ± 0.065	0.213 ± 0.067
	TopER Degree	0.883 ± 0.063	0.061 ± 0.036	2.729 ± 0.930	20.703 ± 4.083	0.110 ± 0.059	0.143 ± 0.044
	TopER Olricci	0.883 ± 0.072	0.162 ± 0.043	1.850 ± 0.481	34.409 ± 6.646	0.160 ± 0.064	0.202 ± 0.055
	TopER Popularity	0.856 ± 0.094	0.178 ± 0.042	2.006 ± 0.608	35.827 ± 5.863	0.211 ± 0.105	0.259 ± 0.070
Persistent Homology	PH Fricci	**1.000 ± 0.000**	0.128 ± 0.045	1.227 ± 0.090	18.835 ± 3.434	0.057 ± 0.029	0.105 ± 0.063
	PH Vietoris Rips	**1.000 ± 0.000**	0.213 ± 0.030	1.264 ± 0.075	29.377 ± 3.326	0.090 ± 0.061	0.139 ± 0.048
	PH Olricci	0.989 ± 0.022	0.176 ± 0.049	0.919 ± 0.047	40.144 ± 6.448	0.081 ± 0.048	0.159 ± 0.046
	PH Laplacian	0.900 ± 0.074	0.034 ± 0.045	2.152 ± 0.176	5.495 ± 1.112	0.021 ± 0.016	0.032 ± 0.020
	PH Degree	0.883 ± 0.091	0.029 ± 0.047	2.237 ± 0.127	5.371 ± 0.788	0.011 ± 0.013	0.037 ± 0.028
	PH Alpha Complex	0.767 ± 0.108	0.067 ± 0.029	2.016 ± 0.321	28.349 ± 6.228	0.124 ± 0.078	0.175 ± 0.048

Table S.4: Accuracies and clustering scores for distinguishing parameter choices in RP graphs.

	Method	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI agglomerative	ARI k-means
Diversity Curves	Shortest Path	**0.978 ± 0.037**	0.576 ± 0.032	0.501 ± 0.038	285.485 ± 47.777	0.448 ± 0.062	**0.674 ± 0.139**
	Shortest Path PCA	**0.978 ± 0.037**	**0.593 ± 0.033**	**0.474 ± 0.041**	**295.427 ± 50.529**	0.458 ± 0.088	0.673 ± 0.141
Diversity Function	Spread Function Shortest Path	0.928 ± 0.070	0.035 ± 0.017	4.557 ± 0.679	6.023 ± 1.646	0.030 ± 0.032	0.021 ± 0.025
Embeddings	NetLSD	0.972 ± 0.037	0.145 ± 0.025	1.385 ± 0.088	32.110 ± 3.923	0.036 ± 0.056	0.382 ± 0.097
	SpectralZoo	0.906 ± 0.066	0.117 ± 0.036	1.678 ± 0.327	31.343 ± 6.921	0.065 ± 0.064	0.229 ± 0.056
	FEATHER	0.861 ± 0.083	0.191 ± 0.029	1.502 ± 0.134	41.949 ± 7.816	0.175 ± 0.073	0.212 ± 0.051
	Graph2Vec	0.622 ± 0.060	0.003 ± 0.002	6.308 ± 0.298	2.655 ± 0.291	0.024 ± 0.044	0.176 ± 0.067
Kernel	Kernel WL MDS	0.972 ± 0.037	0.340 ± 0.036	1.317 ± 0.149	22.752 ± 4.198	0.109 ± 0.119	0.349 ± 0.128
	Kernel Shortest Path	0.967 ± 0.037	0.527 ± 0.042	-	-	**0.510 ± 0.088**	-
	Kernel Shortest Path MDS	0.967 ± 0.027	0.434 ± 0.036	0.990 ± 0.075	40.063 ± 5.645	0.133 ± 0.106	0.469 ± 0.118
	Kernel Core Framework	0.961 ± 0.050	0.345 ± 0.027	-	-	0.244 ± 0.104	-
	Kernel Core Framework MDS	0.950 ± 0.072	0.268 ± 0.035	1.484 ± 0.233	36.632 ± 6.913	0.289 ± 0.129	0.424 ± 0.077
	Kernel WL	0.939 ± 0.063	0.236 ± 0.033	-	-	0.016 ± 0.021	-
	Kernel WL Optimal Assignment	0.889 ± 0.082	0.012 ± 0.004	-	-	0.171 ± 0.065	-
	Kernel Pyramid Match	0.783 ± 0.098	0.038 ± 0.012	-	-	0.010 ± 0.022	-
	Kernel WL Optimal Assignment MDS	0.783 ± 0.098	0.061 ± 0.022	8.316 ± 6.006	8.633 ± 2.437	0.201 ± 0.101	0.189 ± 0.081
	Kernel Pyramid Match MDS	0.767 ± 0.121	0.026 ± 0.018	5.064 ± 0.996	5.336 ± 1.275	0.040 ± 0.027	0.040 ± 0.055
	Kernel Neighbourhood Hash	0.667 ± 0.090	-0.025 ± 0.007	-	-	0.007 ± 0.013	-
	Kernel Neighbourhood Hash MDS	0.611 ± 0.086	-0.050 ± 0.011	11.336 ± 6.653	1.569 ± 0.900	0.006 ± 0.018	0.020 ± 0.029
TopER	TopER All Filtrations	0.872 ± 0.093	0.026 ± 0.037	1.914 ± 0.364	26.117 ± 5.753	0.147 ± 0.042	0.139 ± 0.037
	TopER Closeness	0.783 ± 0.135	0.023 ± 0.038	1.954 ± 0.446	23.042 ± 5.265	0.112 ± 0.050	0.141 ± 0.043
	TopER Popularity	0.706 ± 0.138	-0.025 ± 0.047	2.128 ± 0.522	35.234 ± 8.414	0.158 ± 0.037	0.151 ± 0.034
	TopER Degree	0.678 ± 0.099	-0.028 ± 0.031	2.310 ± 0.930	23.952 ± 5.574	0.085 ± 0.041	0.111 ± 0.037
	TopER Forricci	0.672 ± 0.110	-0.043 ± 0.022	3.328 ± 1.586	11.954 ± 4.146	0.058 ± 0.029	0.060 ± 0.028
	TopER Olricci	0.572 ± 0.075	-0.050 ± 0.022	3.725 ± 1.580	9.520 ± 3.548	0.035 ± 0.029	0.052 ± 0.028
Persistent Homology	PH Alpha Complex	0.894 ± 0.072	0.183 ± 0.031	1.313 ± 0.142	49.927 ± 8.714	0.200 ± 0.090	0.319 ± 0.079
	PH Vietoris Rips	0.889 ± 0.079	0.083 ± 0.031	2.219 ± 0.219	15.271 ± 2.565	0.049 ± 0.040	0.077 ± 0.044
	PH Laplacian	0.889 ± 0.102	-0.092 ± 0.019	3.261 ± 0.337	11.802 ± 2.092	0.015 ± 0.015	0.073 ± 0.037
	PH Fricci	0.828 ± 0.088	-0.045 ± 0.022	2.239 ± 0.203	16.281 ± 3.202	0.035 ± 0.038	0.094 ± 0.035
	PH Degree	0.822 ± 0.096	-0.095 ± 0.019	3.074 ± 0.351	7.049 ± 1.181	0.008 ± 0.012	0.047 ± 0.036
	PH Olricci	0.817 ± 0.096	-0.014 ± 0.026	1.862 ± 0.140	23.389 ± 4.529	0.081 ± 0.039	0.105 ± 0.033

Table S.5: Accuracies and clustering scores for distinguishing parameter choices of RG graphs.

	Method	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI agglomerative	ARI k-means
Diversity Curves	Shortest Path	**0.967 ± 0.044**	0.591 ± 0.032	0.483 ± 0.046	427.861 ± 97.530	0.486 ± 0.056	0.478 ± 0.129
	Shortest Path PCA	**0.967 ± 0.044**	**0.600 ± 0.033**	**0.468 ± 0.046**	**434.844 ± 100.391**	0.486 ± 0.057	0.480 ± 0.134
Diversity Function	Spread Function Shortest Path	0.889 ± 0.066	0.093 ± 0.017	2.660 ± 0.426	20.325 ± 3.851	0.110 ± 0.059	0.126 ± 0.059
Embeddings	FEATHER	0.939 ± 0.072	0.319 ± 0.033	1.173 ± 0.129	87.254 ± 13.845	0.456 ± 0.073	0.489 ± 0.063
	NetLSD	0.922 ± 0.067	0.068 ± 0.025	2.388 ± 0.317	61.887 ± 12.320	0.138 ± 0.057	0.232 ± 0.071
	SpectralZoo	0.906 ± 0.111	0.183 ± 0.032	1.305 ± 0.187	77.332 ± 13.485	0.223 ± 0.145	0.365 ± 0.037
	Graph2Vec	0.578 ± 0.062	0.000 ± 0.002	7.192 ± 0.355	1.483 ± 0.132	0.012 ± 0.029	0.041 ± 0.038
Kernel	Kernel Core Framework MDS	0.939 ± 0.076	0.373 ± 0.030	0.891 ± 0.063	99.746 ± 12.842	0.460 ± 0.151	**0.574 ± 0.146**
	Kernel Core Framework	0.939 ± 0.107	0.386 ± 0.032	-	-	0.486 ± 0.139	-
	Kernel WL Optimal Assignment	0.928 ± 0.056	0.053 ± 0.007	-	-	0.398 ± 0.115	-
	Kernel WL MDS	0.911 ± 0.103	0.226 ± 0.037	3.921 ± 2.201	14.689 ± 4.101	0.025 ± 0.058	0.362 ± 0.130
	Kernel WL Optimal Assignment MDS	0.906 ± 0.075	0.175 ± 0.026	2.739 ± 0.885	22.119 ± 3.766	0.358 ± 0.123	0.350 ± 0.119
	Kernel WL	0.889 ± 0.129	0.185 ± 0.030	-	-	0.001 ± 0.005	-
	Kernel Shortest Path	0.889 ± 0.127	0.428 ± 0.049	-	-	**0.489 ± 0.080**	-
	Kernel Shortest Path MDS	0.861 ± 0.115	0.320 ± 0.043	1.768 ± 0.438	38.574 ± 7.686	0.321 ± 0.198	0.486 ± 0.055
	Kernel Pyramid Match	0.811 ± 0.075	0.040 ± 0.010	-	-	0.028 ± 0.038	-
	Kernel Pyramid Match MDS	0.672 ± 0.088	0.004 ± 0.014	9.754 ± 6.360	3.794 ± 1.051	0.041 ± 0.030	0.025 ± 0.037
	Kernel Neighbourhood Hash	0.578 ± 0.083	-0.014 ± 0.007	-	-	0.002 ± 0.008	-
	Kernel Neighbourhood Hash MDS	0.500 ± 0.119	-0.044 ± 0.007	16.075 ± 8.087	0.807 ± 0.521	-0.004 ± 0.011	-0.005 ± 0.010
TopER	TopER Popularity	0.833 ± 0.129	0.153 ± 0.046	2.013 ± 0.627	33.288 ± 6.509	0.189 ± 0.104	0.270 ± 0.055
	TopER Degree	0.806 ± 0.083	0.048 ± 0.039	2.340 ± 0.874	21.922 ± 4.646	0.099 ± 0.039	0.151 ± 0.039
	TopER All Filtrations	0.800 ± 0.100	0.094 ± 0.039	2.353 ± 0.745	25.906 ± 4.954	0.136 ± 0.075	0.212 ± 0.065
	TopER Closeness	0.800 ± 0.097	0.058 ± 0.039	2.473 ± 1.004	22.667 ± 4.479	0.105 ± 0.048	0.166 ± 0.053
	TopER Forricci	0.800 ± 0.051	0.119 ± 0.041	2.321 ± 0.838	32.456 ± 7.086	0.189 ± 0.078	0.235 ± 0.067
	TopER Olricci	0.650 ± 0.119	0.029 ± 0.043	7.906 ± 5.057	15.313 ± 3.525	0.073 ± 0.045	0.118 ± 0.043
Persistent Homology	PH Vietoris Rips	0.939 ± 0.084	0.114 ± 0.036	1.849 ± 0.119	19.354 ± 2.626	0.044 ± 0.030	0.115 ± 0.040
	PH Olricci	0.900 ± 0.074	0.138 ± 0.050	1.124 ± 0.075	28.997 ± 5.001	0.043 ± 0.036	0.146 ± 0.052
	PH Fricci	0.872 ± 0.079	0.101 ± 0.047	1.568 ± 0.118	13.261 ± 3.103	0.008 ± 0.014	0.097 ± 0.057
	PH Laplacian	0.811 ± 0.106	0.034 ± 0.047	2.030 ± 0.170	6.883 ± 1.267	0.010 ± 0.014	0.043 ± 0.029
	PH Alpha Complex	0.778 ± 0.090	0.105 ± 0.033	1.723 ± 0.272	32.251 ± 6.547	0.130 ± 0.065	0.233 ± 0.078
	PH Degree	0.739 ± 0.108	0.030 ± 0.046	2.535 ± 0.272	4.823 ± 0.962	0.012 ± 0.016	0.029 ± 0.024

Table S.6: Accuracies and clustering scores for distinguishing parameter choices in SBM graphs.

	Method	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI agglomerative	ARI k-means
Diversity Curves	Shortest Path	**0.856 ± 0.079**	0.288 ± 0.036	1.062 ± 0.115	98.923 ± 15.813	0.397 ± 0.064	**0.427 ± 0.068**
	Shortest Path PCA	0.844 ± 0.092	**0.296 ± 0.037**	**1.038 ± 0.113**	**101.116 ± 16.296**	**0.406 ± 0.067**	0.426 ± 0.070
Diversity Function	Spread Function Shortest Path	0.794 ± 0.079	-0.000 ± 0.014	5.739 ± 1.063	3.491 ± 1.383	0.007 ± 0.021	0.002 ± 0.015
Embeddings	NetLSD	0.828 ± 0.080	0.112 ± 0.043	1.899 ± 0.432	20.365 ± 3.643	0.044 ± 0.027	0.095 ± 0.067
	FEATHER	0.656 ± 0.116	0.041 ± 0.024	2.383 ± 0.462	26.831 ± 5.108	0.137 ± 0.059	0.156 ± 0.040
	SpectralZoo	0.633 ± 0.083	0.044 ± 0.035	2.676 ± 0.855	24.065 ± 4.737	0.104 ± 0.053	0.157 ± 0.043
	Graph2Vec	0.622 ± 0.085	0.000 ± 0.002	6.338 ± 0.297	1.882 ± 0.215	0.017 ± 0.043	0.090 ± 0.042
Kernel	Kernel WL Optimal Assignment	0.800 ± 0.075	0.009 ± 0.005	-	-	0.145 ± 0.056	-
	Kernel Core Framework MDS	0.700 ± 0.090	0.029 ± 0.018	3.843 ± 0.664	9.330 ± 2.056	0.037 ± 0.040	0.065 ± 0.035
	Kernel Shortest Path MDS	0.694 ± 0.076	-0.030 ± 0.013	25.418 ± 24.425	0.710 ± 0.570	0.038 ± 0.026	0.034 ± 0.026
	Kernel Core Framework	0.694 ± 0.103	0.029 ± 0.021	-	-	0.033 ± 0.038	-
	Kernel Pyramid Match	0.672 ± 0.091	-0.009 ± 0.007	-	-	-0.001 ± 0.014	-
	Kernel WL MDS	0.672 ± 0.115	-0.036 ± 0.015	25.410 ± 68.230	0.991 ± 0.586	0.002 ± 0.014	0.015 ± 0.019
	Kernel WL	0.661 ± 0.088	-0.017 ± 0.015	-	-	0.009 ± 0.016	-
	Kernel Shortest Path	0.639 ± 0.080	-0.062 ± 0.015	-	-	0.017 ± 0.024	-
	Kernel WL Optimal Assignment MDS	0.639 ± 0.083	-0.001 ± 0.017	4.799 ± 0.956	6.008 ± 1.697	0.101 ± 0.054	0.102 ± 0.046
	Kernel Pyramid Match MDS	0.622 ± 0.092	-0.009 ± 0.011	6.867 ± 2.211	3.509 ± 0.942	0.029 ± 0.030	0.023 ± 0.034
	Kernel Neighbourhood Hash	0.539 ± 0.117	-0.020 ± 0.005	-	-	-0.002 ± 0.010	-
	Kernel Neighbourhood Hash MDS	0.489 ± 0.065	-0.043 ± 0.006	27.497 ± 47.287	0.601 ± 0.419	-0.005 ± 0.009	-0.006 ± 0.010
TopER	TopER Degree	0.700 ± 0.079	-0.072 ± 0.016	5.790 ± 3.233	8.171 ± 3.373	0.027 ± 0.019	0.031 ± 0.022
	TopER Popularity	0.661 ± 0.091	-0.066 ± 0.023	4.008 ± 2.100	12.780 ± 3.919	0.050 ± 0.027	0.060 ± 0.030
	TopER Forricci	0.622 ± 0.078	-0.049 ± 0.022	3.615 ± 1.488	12.333 ± 4.063	0.042 ± 0.026	0.050 ± 0.027
	TopER All Filtrations	0.600 ± 0.065	-0.061 ± 0.019	4.282 ± 1.701	11.334 ± 3.697	0.050 ± 0.028	0.046 ± 0.027
	TopER Olricci	0.594 ± 0.108	-0.061 ± 0.019	4.774 ± 3.093	8.447 ± 3.225	0.035 ± 0.025	0.036 ± 0.025
	TopER Closeness	0.589 ± 0.120	-0.058 ± 0.028	4.114 ± 2.327	13.275 ± 4.404	0.053 ± 0.032	0.071 ± 0.031
Persistent Homology	PH Degree	0.717 ± 0.107	-0.096 ± 0.020	3.451 ± 0.386	6.487 ± 1.265	0.005 ± 0.010	0.056 ± 0.039
	PH Olricci	0.639 ± 0.115	-0.094 ± 0.026	2.729 ± 0.484	12.478 ± 2.966	0.014 ± 0.016	0.081 ± 0.035
	PH Laplacian	0.639 ± 0.143	-0.096 ± 0.018	3.627 ± 0.355	6.482 ± 1.446	0.009 ± 0.013	0.051 ± 0.043
	PH Vietoris Rips	0.611 ± 0.099	-0.085 ± 0.030	4.222 ± 1.024	4.581 ± 1.386	0.010 ± 0.013	0.016 ± 0.019
	PH Alpha Complex	0.606 ± 0.128	-0.053 ± 0.032	4.534 ± 2.876	8.124 ± 3.283	0.025 ± 0.029	0.065 ± 0.032
	PH Fricci	0.600 ± 0.124	-0.097 ± 0.027	3.291 ± 0.536	8.157 ± 1.917	0.005 ± 0.011	0.053 ± 0.033

D.3 Distinguishing between trajectory or cluster structures in single-cell graphs

Single-cell RNA sequencing datasets are often characterised as neighbourhood graphs that connect individual cells based on their molecular similarity. These single-cell graphs can further be characterised as displaying clustering-like or trajectory-like geometries [52]. Clustering-like graphs are dominated by clearly separated clusters of cells. Trajectory-like patterns are characterised by continuous transitions between cell states that form continuously connected trajectories on the single-cell graphs [76]. Knowing which type of geometry a graph displays directly gives actionable insights into which type of downstream analysis and visualisation methods are most appropriate for a given dataset. However, despite its practical relevance, the task of distinguishing between trajectory and clustering structures has not yet been solved. The most recent and to our knowledge only study on this task by Lim and Qiu [52] considers a collection of 169 single-cell datasets, out of which 85 show cluster-like structures and 84 show trajectory-like patterns. These datasets have been collected from a wide range of existing studies on evaluating trajectory inference methods [76], evaluating clustering methods [39, 45], and across a list of further datasets [5, 7–9, 11–13, 15, 16, 20–24, 26, 27, 30, 31, 33, 37, 40, 42, 48–52, 56, 57, 60, 61, 65, 67, 68, 70, 72, 77, 78, 83–85, 87–89, 91, 94, 95, 99]. Out of the final collection of datasets, we consider two versions: First, we consider all 169 datasets altogether. Second, we consider a subset of 27 trajectory-like datasets whose trajectory structure has been verified gold-standard through biological validation [76] as well as 60 cluster-like datasets, which have been annotated as "organs" or "clusters" by Lim and Qiu [52]. Overall, this leaves a subset of 87 graphs, whose annotations are presumed to be less ambiguous.

Previously, Lim and Qiu [52] reach a classification accuracy of around 57% on the collection of all 169 single-cell datasets as reported in their publication. Specifically, Lim and Qiu [52] used a set of 5 geometry-inspired scores to construct a landscape i.e. a shared representation of real single-cell datasets as well as simulated trajectory- or cluster-like datasets. They then use kNN-classification to predict the type of the real datasets based on the simulated examples, leading to a substantial proportion of graphs being misclassified. Given that Lim and Qiu [52] provide exact numbers on how many graphs per category were classified correctly or incorrectly in Figure 6.c of their manuscript, we use these numbers to simulate the performance of their Landscape approach across 5-fold stratified cross validation and report the estimated accuracies in Table S.7. Overall, we note that this reaches only about 57% accuracy. We aim to improve upon these results in our own study by investigating this classification task across a broader range of baseline methods.

Following Lim and Qiu [52], we download their dataset of processed single-cell datasets, and the set of scores they computed for each dataset, which have been made available with their codebase[10] and their publication. Preprocessing and quality control for the RNA sequencing datasets has been applied as described by Saelens et al. [76]. Lim and Qiu [52] further used PCA to reduce the dimensionality of each dataset to 20 principal components. Given the substantial size differences between datasets, ranging between around 19953 and 56 observations, we apply geometric sketching to subsample each dataset to at most 200 cells [32]. Further, we convert each dataset into a graph by connecting each cell to its 10 closest neighbours. Figure 5 then shows representative examples of four of these processed single-cell graphs, two trajectory-like graphs (A: cellbench-SC2_luyitian, GSE118767, simulated trajectory-like human cells [76, 84]; B: aging-hsc-young_kowalczyk, GSE59114, hematopoietic stem cell of mice [42, 76]), as well as two cluster-like graphs (C: mouse-cell-atlas-combination-5, cells from a mouse tissue atlas [30, 76]; D: GSE82187, mouse brain cells [22, 39]). Across these examples, we see that cluster-like graphs have distinct communities, while trajectory-like graphs display continuous transitions between cells. Our aim is now to distinguish these two types of graph through their structure.

We input these graphs into graph embedding models, or kernel methods. Further, we compute diversity curves with shortest-path distances, diffusion distances, or Euclidean distances between the scaled node features. The resulting mean diversity curves per group and their standard deviation are shown in Figure S.7. These diversity curves demonstrate that cluster-like single-cell graphs have on average higher structural diversity as measured by diversity curves, which use adjacency-based distances, such as shortest-path or diffusion distances. This is motivated by the fact that clusters of cells in these graphs are more distinct and more well separated structurally. These trends carry over to the PCA plot in Figure 5, which is computed from the standard scaled diversity curves using shortest-path as well as feature-based distances. We further match each of the four example graphs

[10] Made available at https://github.com/pqiu/Quantifying_clusterness_trajectoriness with the publication of their paper [52].

Table S.7: Model performance metrics (mean ± std) at distinguishing between trajectory- and cluster-like single-cell graphs using kNN-classification across 5-fold stratified cross validation.

	Method	Accuracy - All Graphs	AUROC - All Graphs	Accuracy - Gold	AUROC - Gold
Diversity Curves	Diversity Curves All Metrics	**0.766 ± 0.062**	**0.810 ± 0.068**	**0.860 ± 0.062**	0.824 ± 0.068
	Shortest Path + Feature Dist.	0.759 ± 0.077	0.808 ± 0.081	**0.860 ± 0.077**	0.831 ± 0.081
	Diffusion Dist.	0.743 ± 0.061	0.786 ± 0.065	0.807 ± 0.061	0.795 ± 0.065
	Feature Dist.	0.721 ± 0.075	0.759 ± 0.089	0.843 ± 0.075	0.809 ± 0.089
	Shortest Path	0.682 ± 0.095	0.723 ± 0.115	0.773 ± 0.095	0.757 ± 0.115
Embeddings	NetLSD	0.731 ± 0.047	0.802 ± 0.057	0.823 ± 0.047	**0.848 ± 0.057**
	FEATHER	0.715 ± 0.071	0.771 ± 0.085	0.830 ± 0.071	0.824 ± 0.085
	SpectralZoo	0.675 ± 0.074	0.734 ± 0.095	0.821 ± 0.074	0.799 ± 0.095
	Graph2Vec	0.622 ± 0.074	0.648 ± 0.088	0.741 ± 0.074	0.790 ± 0.088
Scores [52]	All Scores	0.716 ± 0.053	0.760 ± 0.061	0.804 ± 0.053	0.797 ± 0.061
	PH Score	0.697 ± 0.058	0.747 ± 0.066	0.829 ± 0.058	0.830 ± 0.066
	Reachability Score	0.632 ± 0.070	0.667 ± 0.087	0.727 ± 0.070	0.709 ± 0.087
	Vector Norm Score	0.626 ± 0.074	0.671 ± 0.102	0.694 ± 0.074	0.668 ± 0.102
	Diffusion Distance Score	0.520 ± 0.089	0.516 ± 0.100	0.611 ± 0.089	0.600 ± 0.100
	Landscape	0.570 ± 0.025	0.586 ± 0.037	-	-
	Ripley's K Score	0.532 ± 0.079	0.559 ± 0.088	0.598 ± 0.079	0.458 ± 0.088
Kernel	Kernel Pyramid Match	0.685 ± 0.092	0.721 ± 0.081	0.784 ± 0.092	0.757 ± 0.081
	Kernel WL	0.652 ± 0.079	0.699 ± 0.099	0.785 ± 0.079	0.773 ± 0.099
	Kernel Shortest Path	0.649 ± 0.085	0.690 ± 0.104	0.782 ± 0.085	0.770 ± 0.104
	Kernel Core Framework	0.649 ± 0.085	0.690 ± 0.104	0.782 ± 0.085	0.770 ± 0.104
	Kernel WL Optimal Assignment	0.600 ± 0.062	0.626 ± 0.083	0.679 ± 0.062	0.653 ± 0.083
	Kernel Neighborhood Hash	0.568 ± 0.082	0.588 ± 0.083	0.704 ± 0.082	0.667 ± 0.083
TopER	TopER All Filtrations	0.624 ± 0.086	0.669 ± 0.121	0.672 ± 0.086	0.649 ± 0.121
	TopER Olricci	0.586 ± 0.058	0.607 ± 0.073	0.649 ± 0.058	0.659 ± 0.073
	TopER Forricci	0.581 ± 0.053	0.600 ± 0.064	0.670 ± 0.053	0.610 ± 0.064
	TopER Popularity	0.516 ± 0.079	0.543 ± 0.090	0.662 ± 0.079	0.563 ± 0.090
	TopER Closeness	0.505 ± 0.069	0.520 ± 0.076	0.624 ± 0.069	0.476 ± 0.076
	TopER Degree	0.503 ± 0.090	0.521 ± 0.089	0.630 ± 0.090	0.480 ± 0.089
Node Features	Mean Pooling	0.619 ± 0.091	0.652 ± 0.103	0.661 ± 0.091	0.614 ± 0.103
	Standard Scaling + Mean Pooling	0.559 ± 0.068	0.573 ± 0.006	0.553 ± 0.068	0.585 ± 0.060

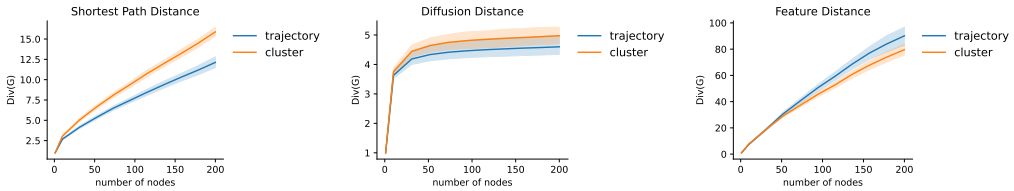

Figure S.7: Mean diversity curves per trajectory type computed using shortest-path distances, diffusion distances, or Euclidean distances based on standard scaled node features.

A to D to their corresponding representation in the PCA visualisation. Overall, this demonstrates visually that diversity curves distinguish the geometry of single-cell graphs.

Beyond the results reported in our main experiments, we further provide comparisons with feature-based baselines. Specifically, we apply mean pooling across the node features of the downsampled single-cell datasets as well as on the standard scaled version of these features. Comparing graph-based embeddings with these feature baselines then helps to assess to what extent the construction of neighbourhood graphs aids with distinguishing the geometries of single-cell datasets.

Table S.7 reports kNN-classification results for each method across 5-times repeated 5-fold stratified cross-validation, where 64% of the data is used for training, 16% for validation, and 20% for testing. The number of k nearest neighbours is tuned by grid search on the validation dataset in a range of $\{1, 3, 5, 7, 9\}$. We use a standard scaling transform on the diversity curves as this transformation on the training set helps highlight differences across graph sizes. Finally, Table S.7 shows the performance of each method, including the geometry scores by Lim and Qiu [52], in terms of both classification accuracies and AUROC scores.

D.4 Characterising molecular graph datasets

To describe structural similarities between enzyme classes, we download the `PROTEINS` and `ENZYMES` datasets [6, 17], which are available under the TUDataset [59] collection through PyTorch Geometric

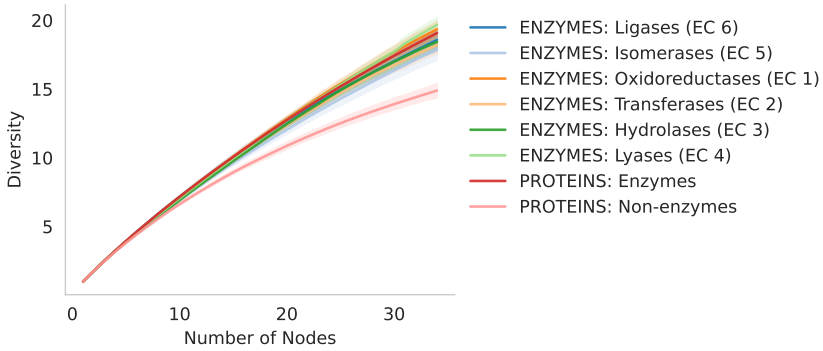

Figure S.8: Mean diversity curves per class in the PROTEINS and ENZYMES datasets.

(available under an MIT Licence). We map the classes from the PROTEINS graphs to the labels "enzymes" or "non-enzymes" by checking with the original publications [6, 17] and verifying the number of graphs in each class matches what is described by Borgwardt et al. [6]. Classes in ENZYMES are characterised by their Enzyme Commission (EC) number. We then compare the class of proteins that are not enzymes, to the classes of proteins that are enzymes across both datasets. To do so, we compute diversity curves using shortest-path distances on the graphs across node ranges from 1 to 34 nodes. The mean curves per class are visualised in Figure S.8 and highlight that non-enzymes have notably lower structural diversity on average than enzyme graphs. To verify this, we implement the statistical test for equality in the mean diversity curves as described in Section 4.4. This case study demonstrates that diversity curves can be used for statistical testing, which highlights their utility for data-driven and uncertainty-aware analysis. Further, diversity curves directly allow for comparisons between differently sized graphs and graph distributions across varying cardinalities. Based on this, we believe there is promising potential for further applications to describe, explore, and compare the structural properties of graph datasets, both to describe the quality of graph datasets [14], as well as towards evaluating structural similarities for transfer learning tasks [34].

D.5 Distinguishing geometric shapes

To test whether our methods encode the inherent structure of data, we use the MANTRA[11] dataset [3], which has been proposed to benchmark the capability of geometric and topological deep learning models to distinguish between combinatorial triangulations of manifolds. Table S.8 specifically shows the number of triangulations of 2-manifolds in the MANTRA dataset, out of which triangulations of the Klein bottle (K), torus (T^2), sphere (S^2), and real projective plane (RP^2) are associated with explicit homeomorphism types that characterise their shape. For our main experiments, we address the task of distinguishing between these four types.

We follow the experimental setup by Ballester et al. [3] to evaluate each classification model across 5-fold stratified cross validation, taking 20% of the data for testing, 16% for validation, and 64% for training. In this manner, we train Support Vector Machines with radial basis function kernels using default parameters as implemented in scikit-learn (version 1.8.0) unless mentioned otherwise. For hyperparameter tuning, we use randomised grid search to select the cost parameter C in $[0.01, 100]$ and gamma in $[0.0001, 100]$ with log-uniform probabilities across 20 random iterations and 3-fold cross validation on the validation set. We let our models predict probabilities to report the one-vs-rest multi-class AUROC score on the test set, and further report the test accuracies to compare with the results by Ballester et al. [3]. All performance scores for the Cellular Transformer (CT) are cited directly from Table 11 in Ballester et al. [3], where this model is the best performing architecture at the intended task.

To compute diversity curves, we use edge contraction for coarsening as described throughout our paper. But we also note that some triangulations in MANTRA only have very few nodes making direct comparisons with larger graphs difficult. Therefore, we also employ upsampling operations via upsampling each 3-clique in the graph, or each triangle in the triangulation as described in

[11] https://github.com/aidos-lab/MANTRA available under a BSD-3-Clause Licence.

Table S.8: The number of triangulations of 2-manifolds per shape for all classes with more than 100 objects in the MANTRA dataset.

Surface	$\#^4\mathrm{RP}^2$	$\#^3\mathrm{RP}^2$	$\#^5\mathrm{RP}^2$	Klein bottle	T^2	RP^2	$\#^6\mathrm{RP}^2$	$\#^2T^2$	S^2
Number	13695	11917	7053	4657	2247	1365	1022	868	308

Table S.9: Model performance (mean ± std) at distinguishing triangulations of 2-manifolds across stratified 5-fold CV.

Method	Version	Accuracy	AUROC
Diversity Curve	$\mathcal{T}$	1.000 ± 0.001	1.000 ± 0.000
Diversity Curve	G	0.928 ± 0.001	0.975 ± 0.002
Optimal Classifier	G	0.948 ± 0.000	0.941 ± 0.000
Diversity Curve	$\mathcal{T}_{\text{up}}$	0.3882 ± 0.006	0.841 ± 0.014

Section B.1.1. Specifically, we uniformly select triangles making sure that each triangle is picked at least once before a previously upsampled triangle could be selected again, and subdivide the selected triangle by adding a central vertex and connecting it to all corners. This way, we uniformly subdivide the surface of the meshes ($\mathcal{T}$), or the 3-cliques in the graphs (G).

We report three versions of our results:

G: First, to train on graphs, denoted by G, we convert each triangulation into their one-skeleton. Then we compute diversity curves based on shortest-path distances on these graphs, for cardinalities between 1 and 15, the maximum number of nodes in the graphs. Note that for this version of the task, tori and Klein bottles will have identical 1-skeleton graphs, and thus be indistinguishable by their graph structure alone [3]. Based on this observation, we choose to report the performance of an optimal or dummy classifier in Table 4, whose performance we simulate by assuming that all classes other than tori get classified correctly with probability 1, while the triangulations of a torus get confidently misclassified as Klein bottles again with probability 1, which we use to simulate both the accuracy and the multi-class AUROC of this classifier baseline.

$\mathcal{T}$: Second, we input the triangulations themselves into our model, denoted by $\mathcal{T}$, which include information on nodes, edges, and faces allowing us to integrate higher order features into the predictions. Specifically, we compute heat kernel distances between faces to compute diversity curves on these higher order structures for cardinalities between 3 and 15. We then concatenate these curves with the previously computed graph-based diversity curves, as inputs for our prediction models.

$\mathcal{T}_{\text{up}}$: Further, we apply barycentric subdivision to subdivide each mesh. As described by Ballester et al. [3], we then aim to address the task of training our models on the original dataset, but testing it on these subdivided meshes. Effectively, models that can solve this challenge have learned the intrinsic shape of each object while being robust to differences in scale and cardinality. This version of the task is denoted by $\mathcal{T}_{\text{up}}$. We compute diversity curves using both shortest-path distances on the graphs, and higher order heat kernel distances on the faces. For computing these higher-order distances we set $t = 20$ and $k = 2$ to compute a metric between triangles. Further, we use triangular upsampling to compute diversity curves for node sizes from 1 to 15 with a step size of one, and 20 to 60 with a step size of 5, extending the evaluation scales to capture trends at higher cardinalities in a size-aware manner. The final results for each version of the task are reported in Table 4. Further, in this scenario we apply standard scaling to the test and the training set independently to specifically account for domain-shift between the original shapes and upsampled shapes as we expect their diversity curves to follow the same overall trends but across varying ranges.

Finally, we extend the three tasks above to a larger set of triangulations in MANTRA. Instead of just taking the four shapes with known homeomorphism types, we also consider all other classes of 2-manifolds in MANTRA that consist of more than 100 elements. Results for this extended version are reported in Table S.9.

D.6 Expressivity

The BREC[12] dataset [92] consists of 400 pairs of graphs, which are used to test the expressivity of graph learning methods. All pairs of graphs are 1-WL indistinguishable, and 270 remain 3-WL indistinguishable [92]. We first collect results that have been reported in existing literature and summarise how many pairs of graphs each method can distinguish.

Wang and Zhang [92] report a range of graph learning models, for which we report the three best performing GNNs, I2-GNN, KP-GNN, and GSN, as well as Graphormer as a notably worse-performing transformer baseline. Relevant to our use of edge contraction pooling for comparing graphs, further work by Lachi et al. [46] has investigated under which conditions graph pooling operators do not only maintain but also improve expressivity. Their results prove that edge-based pooling in particular increases expressivity. In contrast, node-based pooling operators that do not incorporate any information beyond WL into their pooling choices, such as DiffPool which clusters nodes by utilising MPNNs, cannot improve expressivity. In fact, results by Lachi et al. [46] show DiffPool distinguishes no pair of graphs in BREC correctly, whereas EdgePool as well as their own edge-based pooling operator, EXpressive Pooling (XP), both improve expressivity on BREC. These results are summarised in Table S.10 as cited from the original publications [46, 92].

Further, we report results by Ballester and Rieck [2] on the performance of alternative geometric methods, namely persistent homology (PH) at distinguishing graphs from BREC. Specifically, we report the performance of different filtrations when filtering through all 2-simplices in the graphs, which represent 3-cliques, and computing persistent homology up to dimension 2, which tracks the evolution of connected components, circles, and two-dimensional voids across the filtration. See Ballester and Rieck [2] for a more detailed description.

Motivated by our theoretical results on the expressivity of diversity curves, we investigate the performance of our methods on BREC in practice. First, we check if spread itself can distinguish each pair of graphs by testing whether their difference in spread is above a given error tolerance [81], which we set to 10^{-5}. We try this for shortest-path, diffusion, and heat kernel distances on the graph, and note that the latter two metric choices can distinguish a larger set of graphs based on their structural diversity alone. We reason this is likely because both are computed from the normalised graph Laplacian, which can be more sensitive to subtle changes in graph structure, than shortest-path distances. Nevertheless, even structural diversity based on shortest-path distances can already distinguish 1-WL indistinguishable graphs.

Next, we extend our experiment to investigate whether edge contraction pooling can improve expressivity. To do so, we check each pair of graphs that cannot be distinguished through their spread alone, and iterate through all possible choices for contracting one edge, and compute the average diversity curve at $n - 1$ for each graph, where n denotes the cardinality. Based on this, we again check if the difference in diversity curves at $n - 1$ is larger than the error tolerance 10^{-5}. Across different choices of graph metric, we observe that edge contraction pooling enables diversity curves to distinguish more pairs of graphs than structural diversity alone. In particular, diversity curves based on shortest-path distances on the graph benefit from this coarsening approach.

For each comparison, we further check a reliability set to ensure that within the chosen error tolerance, pairs of isomorphic graphs are not distinguished. This empirical check holds across all comparisons verifying the validity of our results. From a theoretical perspective, we know that isomorphic graphs have the same deck of edge contractions [71], and thus have the same average diversity across all possible edge contraction choices as further described in Section C.4. Note that the expressivity results for diversity curves presented in Table S.10 represent a best case scenario where the sequences of coarsened graphs for which diversity is computed contain all possible edge contractions. This allows us to compare the difference in diversity curves exactly rather than requiring a more complex statistical approach to account for randomness in the graph embeddings, which necessitates more stringent error control and can be unreliable [92].

For further comparison, we repeat the same procedure above on deterministic graph embedding methods, NetLSD and FEATHER, by computing the embedding vector for each graph and testing if the L^2-norm between embedding vectors is above the error threshold 10^{-5}.

[12] https://github.com/GraphPKU/BREC available under an MIT licence.

Table S.10: Number of pairs from the BREC dataset distinguished via structural diversity or diversity curves. Compared to representative results from [2, 46, 81, 92].

Method	Details	Basic (60)	Regular (140)	Extension (100)	CFI (100)	All (400)
3 WL	(from [92])	60 (100.0%)	50 (35.7%)	100 (100.0%)	60 (60.0%)	270 (67.5%)
1 WL	(from [81])	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
I2-GNN	(from [92])	60 (100.0%)	100 (71.4%)	100 (100.0%)	21 (21.0%)	281 (70.2%)
KP-GNN	(from [92])	60 (100.0%)	106 (75.7%)	98 (98.0%)	11 (11.0%)	275 (68.8%)
GSN	(from [92])	60 (100.0%)	99 (70.7%)	95 (95.0%)	0 (0.0%)	254 (63.5%)
Graphormer	(from [92])	16 (26.7%)	12 (8.6%)	41 (41.0%)	10 (10.0%)	79 (19.8%)
CurvPool	(from [46])	35 (58.3%)	65 (46.4%)	53 (53.0%)	4 (4.0%)	157 (39.3%)
XP	(from [46])	48 (80.0%)	3 (2.1%)	72 (72.0%)	1 (1.0%)	124 (31.0%)
EdgePool	(from [46])	13 (21.7%)	0 (0.0%)	10 (10%)	0 (0.0%)	23 (5.8%)
DiffPool	(from [46])	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)	0 (0.0%)
PH Laplacian	(from [2])	60 (100%)	49 (35%)	100 (100%)	6 (6%)	216 (54%)
PH Ollivier–Ricci Curvature	(from [2])	60 (100%)	54 (38.6%)	92 (92%)	3 (3%)	208 (52%)
PH Forman–Ricci Curvature	(from [2])	59 (98%)	50 (35.7%)	59 (59%)	3 (3%)	172 (43%)
PH Degree Filtration	(from [2])	47 (78%)	39 (27.9%)	26 (26%)	3 (3%)	116 (29%)
PH Vietoris-Rips	(from [2])	31 (52%)	39 (27.9%)	11 (11%)	3 (3%)	84 (21%)
NetLSD	[75, 90]	60 (100.0%)	50 (35.7%)	100 (100.0%)	3 (3.0%)	213 (53.2%)
FEATHER	[74, 75]	52 (86.7%)	46 (32.9%)	5 (5.0%)	0 (0.0%)	103 (25.8%)
Div(G)	heat kernel distance	60 (100.0%)	50 (35.7%)	92 (92.0%)	3 (3.0%)	205 (51.3%)
Div(G)	diffusion distance	60 (100.0%)	49 (35.0%)	84 (84.0%)	3 (3.0%)	196 (49.0%)
Div(G)	shortest path metric	16 (26.7%)	13 (9.3%)	41 (41.0%)	6 (6.0%)	76 (19.0%)
DivCurve($\mathcal{G}_{\{n-1,n\}}$)	heat kernel distance	60 (100.0%)	65 (46.4%)	96 (96.0%)	3 (3.0%)	224 (56.0%)
DivCurve($\mathcal{G}_{\{n-1,n\}}$)	diffusion distance	60 (100.0%)	58 (41.4%)	94 (94.0%)	3 (3.0%)	215 (53.8%)
DivCurve($\mathcal{G}_{\{n-1,n\}}$)	shortest path metric	60 (100.0%)	48 (34.3%)	87 (87.0%)	8 (8.0%)	203 (50.8%)

Finally, Table S.10 reports how well each of the baseline methods or our diversity curves can distinguish pairs of graphs from BREC. Overall, we find that diversity curves perform better or competitively to existing geometry-aware baseline methods, such as persistent homology, or graph embedding methods, such as NetLSD and FEATHER. In fact, results improve upon any previously reported performance of edge pooling layers, and are better than certain baselines reported for notably more parametrised models, such as Graphormer. Overall, we thus see diversity curves improve the expressivity of diversity through edge contraction pooling in practice, while ensuring competitive performance on BREC.

D.7 Computational efficiency

To investigate computational costs empirically, we compare the runtime of our method to relevant baseline methods. We generate 30 Erdős-Rényi graphs $G_{n,p}$ for $n \in \{2^i | i \in \{3, \dots, 13\}\}$ and $p = 1/n$. For each method, we average the runtime over 10 runs. We choose to evaluate diversity curves at the cardinalities $\mathcal{I} = \{20 \cdot i + 10 | 0 \leq i \leq 4\}$, hence the maximal cardinality at which we compute the diversity is $n_{max} = 90$. We include a version of the diversity curves computation with improvements in the computational efficiency. Concretely, the edge dropping is implemented more efficiently using a random permutation of the edges rather than first generating and then sorting the edge scores, and

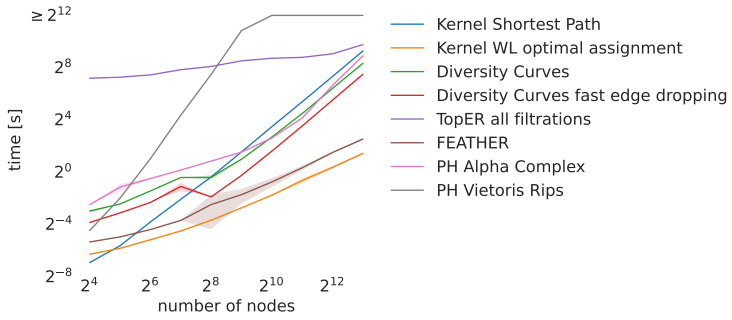

Figure S.9: Empirical runtimes of computing the diversity curves and baselines on ER graphs with increasing number of node sizes in a log-log-plot.

edges are collapsed as long as possible, stopping as soon as the minimal cardinality of the evaluation scales is reached. This speed-up is also discussed in Section C.1.

The results can be seen in Figure S.9. While certain variations of kernel methods and persistent homology are faster to compute than the diversity curves for this experimental setup, we note that we expect kernels to scale worse with a larger dataset size. Further, since our configuration of the diversity curves relies on the shortest-path distance in the graph, when comparing with other shortest-path based methods, such as persistent homology using the Vietoris-Rips filtration, we note that our method outperforms this baseline especially at larger graph sizes. Hence, we conclude that diversity curves can still be reasonably efficiently computed for larger graphs, and are slightly faster than for example the shortest-path kernel or persistent homology using the alpha complex. Because coarsening is an integral component of our framework, computational costs can be efficiently reduced by preprocessing graphs to smaller sizes, or evaluating diversity curves at fewer evaluation scales or lower node numbers. To compute persistent homology for this experiment, we build a clique complex of the graph by adding all 2-simplices, which correspond to 3-cliques in the graph, and compute the persistent homology on the filtrations of the clique complex up to dimension 2. This is the same setting that we used in the persistent homology baselines in the experiment from Section 4.2. The analysis of the empirical runtimes was performed on a compute node with ×2 CPU AMD EPYC 7763 64-CORE PROCESSOR and 1 TiB DRAM.

In addition, we repeat the experiment on distinguishing graphs from different simulated distributions from Table 1 with larger graphs of cardinalities ranging from 16 to 4096, where we generate 5 graphs from each cardinality. We use the fast edge dropping method described above and additionally for all methods chunk the graphs and process the chunks in parallel. We choose parameters, so that the graphs remain connected and evaluate the diversity curves at cardinalities $\mathcal{I} = \{16, 30, 50, 70, 90\}$. Table S.11 shows accuracies and silhouette scores as well as runtimes for diversity curves against NetLSD, FEATHER and SpectralZoo. The other baseline methods encountered computational bottlenecks, reaching memory or runtime errors and are thus not reported. Overall, we find that diversity curves retain competitive performance at distinguishing graph distributions while remaining a fast alternative, even at more vastly varying ranges of graph sizes. Particularly, the use of coarsening and a smart selection of evaluation scales our methods remain scalable on larger graphs.

Table S.11: Comparison of methods across node ranges.

Node Range	Method	Accuracy	Silhouette Score	Time
2^4–2^8	Diversity Curve (ours)	**0.74 ± 0.21**	**0.08 ± 0.06**	6.0s ± 0.0s
	NetLSD	0.60 ± 0.10	−0.12 ± 0.05	**1.1s ± 0.1s**
	FEATHER	0.57 ± 0.09	−0.07 ± 0.02	3.5s ± 0.0s
	SpectralZoo	0.54 ± 0.10	−0.19 ± 0.04	1.2s ± 0.0s
2^4–2^9	Diversity Curve (ours)	**0.74 ± 0.19**	**−0.02 ± 0.06**	9.5s ± 0.1s
	NetLSD	0.73 ± 0.19	−0.15 ± 0.05	9.1s ± 0.0s
	FEATHER	0.60 ± 0.11	−0.06 ± 0.01	31.4s ± 0.0s
	SpectralZoo	0.60 ± 0.12	−0.21 ± 0.04	**2.1s ± 0.1s**
2^4–2^{10}	Diversity Curve (ours)	0.72 ± 0.21	−0.08 ± 0.05	37.8s ± 0.2s
	NetLSD	**0.74 ± 0.11**	−0.17 ± 0.05	8m 7s ± 2.8s
	FEATHER	0.69 ± 0.09	**−0.05 ± 0.01**	4m 4s ± 0.8s
	SpectralZoo	0.66 ± 0.11	−0.23 ± 0.03	**5.3s ± 0.1s**
2^4–2^{11}	Diversity Curve (ours)	0.58 ± 0.34	−0.13 ± 0.05	2m 42s ± 0.5s
	NetLSD	**0.78 ± 0.19**	−0.18 ± 0.05	27m 28s ± 16.7s
	FEATHER	0.69 ± 0.13	**−0.05 ± 0.01**	35m 53s ± 3.2s
	SpectralZoo	0.63 ± 0.19	−0.23 ± 0.04	**17.6s ± 0.1s**
2^4–2^{12}	Diversity Curve (ours)	0.67 ± 0.31	**−0.17 ± 0.05**	11m 30s ± 0.4s
	NetLSD	**0.81 ± 0.19**	−0.21 ± 0.04	1h 23m ± 13.9s
	FEATHER	–	–	> 2h
	SpectralZoo	0.67 ± 0.19	−0.24 ± 0.04	**1m 15s ± 0.2s**

D.8 Ablation studies

In the following, we provide an extended overview of the ablation studies and additional experiments performed to clarify the impact of parameter, metric, evaluation scale, and coarsening method choices on the evaluation and performance of diversity curves in practice.

D.8.1 Choice of distance and edge scores

We perform an ablation study to investigate the choice of our parameters, specifically the choice of graph distance to compute the spread and the choice of the edge scoring method. With the same experimental setup for distinguishing parameter choices in the ER, RP, RG, and SBM graphs as described in Section D.2, we compute the diversity curves for the spread with shortest-path distance, diffusion distance, and resistance distance. Furthermore, for each choice of distance we vary the edge scoring method between random edge scores, edge scores based on the difference in spread, and an approximation of the edge scores based on the spread which approximates the distance between nodes after edge contraction by updating the distances on the original graph by taking the minimum distance to the original nodes [54]. We report all the accuracies and clustering metrics in Table S.12, Table S.15, Table S.14, and Table S.13. We further visualise the different silhouette scores in Figure S.10. We find that over our chosen random graph models, the shortest-path distance performs the best and the choice of edge scoring does not have any significant impact, which is why we decided to opt for random edge scoring in order to reduce the computational time complexity.

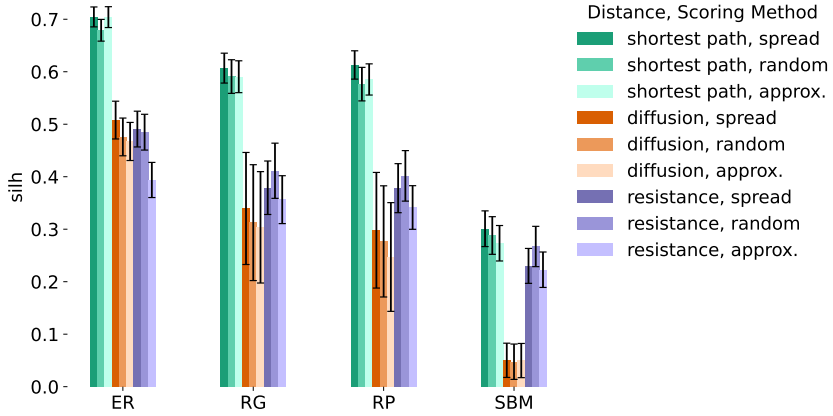

Figure S.10: Silhouette scores at distinguishing different parameter choices, varying the distance used to compute the spread and the edge scoring method.

Table S.12: Accuracies and clustering scores for distinguishing parameter choices for ER graphs.

Distance	Score	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI agglomerative	ARI k-means
shortest path	spread	**1.000 ± 0.000**	**0.704 ± 0.019**	0.313 ± 0.017	**288.211 ± 57.504**	**0.491 ± 0.048**	0.442 ± 0.014
	random	**1.000 ± 0.000**	0.679 ± 0.021	0.346 ± 0.018	272.892 ± 54.881	0.486 ± 0.053	**0.444 ± 0.056**
	approx.	**1.000 ± 0.000**	0.704 ± 0.020	**0.312 ± 0.019**	282.266 ± 57.066	0.491 ± 0.047	0.441 ± 0.015
diffusion	spread	**1.000 ± 0.000**	0.508 ± 0.036	0.552 ± 0.056	69.180 ± 15.427	0.190 ± 0.091	0.271 ± 0.061
	random	**1.000 ± 0.000**	0.476 ± 0.036	0.560 ± 0.052	67.561 ± 15.040	0.194 ± 0.102	0.270 ± 0.063
	approx.	**1.000 ± 0.000**	0.467 ± 0.036	0.557 ± 0.049	69.157 ± 15.465	0.191 ± 0.094	0.271 ± 0.061
resistance	spread	0.994 ± 0.017	0.491 ± 0.034	0.579 ± 0.036	163.684 ± 34.156	0.390 ± 0.068	0.375 ± 0.033
	random	**1.000 ± 0.000**	0.485 ± 0.034	0.569 ± 0.034	146.921 ± 30.415	0.376 ± 0.074	0.365 ± 0.034
	approx.	**1.000 ± 0.000**	0.394 ± 0.033	0.791 ± 0.053	158.064 ± 33.425	0.368 ± 0.069	0.361 ± 0.033

Table S.13: Accuracies and clustering scores for distinguishing parameter choices for RP graphs.

Distance	Score	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI agglomerative	ARI k-means
shortest path	spread	**0.978 ± 0.027**	**0.613 ± 0.027**	0.505 ± 0.037	**285.518 ± 36.276**	**0.480 ± 0.153**	**0.789 ± 0.188**
	random	**0.978 ± 0.037**	0.576 ± 0.032	**0.501 ± 0.038**	285.485 ± 47.777	0.448 ± 0.062	0.674 ± 0.139
	approx.	**0.978 ± 0.037**	0.585 ± 0.030	0.532 ± 0.043	246.763 ± 34.457	0.433 ± 0.135	0.740 ± 0.187
diffusion	spread	0.961 ± 0.056	0.298 ± 0.110	0.829 ± 0.155	55.944 ± 13.154	0.218 ± 0.041	0.218 ± 0.041
	random	0.967 ± 0.037	0.277 ± 0.106	0.846 ± 0.174	62.807 ± 15.787	0.218 ± 0.041	0.218 ± 0.041
	approx.	0.972 ± 0.028	0.247 ± 0.104	0.884 ± 0.168	55.656 ± 13.187	0.218 ± 0.041	0.218 ± 0.041
resistance	spread	0.972 ± 0.037	0.378 ± 0.047	0.777 ± 0.060	146.244 ± 27.256	0.346 ± 0.051	0.371 ± 0.045
	random	0.967 ± 0.044	0.402 ± 0.048	0.728 ± 0.058	181.881 ± 39.252	0.364 ± 0.062	0.384 ± 0.047
	approx.	0.956 ± 0.069	0.341 ± 0.042	0.847 ± 0.049	141.658 ± 28.112	0.346 ± 0.042	0.359 ± 0.053

Table S.14: Accuracies and clustering scores for distinguishing parameter choices for RG graphs.

Distance	Score	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI agglomerative	ARI k-means
shortest path	spread	**0.983 ± 0.036**	**0.607 ± 0.028**	**0.445 ± 0.037**	393.501 ± 80.612	0.485 ± 0.051	**0.486 ± 0.120**
	random	0.967 ± 0.044	0.591 ± 0.032	0.483 ± 0.046	**427.861 ± 97.530**	**0.486 ± 0.056**	0.478 ± 0.129
	approx.	0.972 ± 0.037	0.591 ± 0.030	0.472 ± 0.042	381.289 ± 83.355	0.478 ± 0.053	0.462 ± 0.114
diffusion	spread	0.956 ± 0.042	0.339 ± 0.107	0.784 ± 0.184	124.151 ± 26.770	0.292 ± 0.107	0.357 ± 0.036
	random	0.950 ± 0.046	0.313 ± 0.110	0.845 ± 0.194	124.600 ± 27.846	0.287 ± 0.110	0.352 ± 0.037
	approx.	0.950 ± 0.046	0.304 ± 0.106	0.863 ± 0.188	124.430 ± 26.946	0.292 ± 0.107	0.357 ± 0.035
resistance	spread	0.950 ± 0.058	0.379 ± 0.051	0.798 ± 0.085	219.894 ± 57.742	0.439 ± 0.080	0.407 ± 0.029
	random	0.967 ± 0.044	0.411 ± 0.053	0.757 ± 0.087	254.085 ± 69.380	0.460 ± 0.068	0.412 ± 0.027
	approx.	0.944 ± 0.056	0.356 ± 0.046	0.840 ± 0.071	218.943 ± 57.164	0.438 ± 0.081	0.407 ± 0.028

Table S.15: Accuracies and clustering scores for distinguishing parameter choices of SBM graphs

Distance	Score	Accuracy	Silhouette Score	Davies-Bouldin	Calinski-Harabasz	ARI agglomerative	ARI k-means
shortest path	spread	**0.861 ± 0.062**	**0.301 ± 0.034**	1.122 ± 0.130	94.305 ± 13.963	0.371 ± 0.117	**0.432 ± 0.076**
	random	0.856 ± 0.079	0.288 ± 0.036	1.062 ± 0.115	**98.923 ± 15.813**	**0.397 ± 0.064**	0.427 ± 0.068
	approx.	0.861 ± 0.062	0.273 ± 0.034	1.154 ± 0.130	87.747 ± 13.267	0.376 ± 0.108	0.421 ± 0.074
diffusion	spread	0.783 ± 0.072	0.050 ± 0.033	2.158 ± 0.446	25.955 ± 5.913	0.153 ± 0.058	0.136 ± 0.035
	random	0.822 ± 0.054	0.048 ± 0.034	2.156 ± 0.457	25.918 ± 6.075	0.162 ± 0.047	0.160 ± 0.045
	approx.	0.783 ± 0.072	0.050 ± 0.033	2.161 ± 0.444	25.843 ± 5.908	0.152 ± 0.056	0.136 ± 0.035
resistance	spread	0.822 ± 0.060	0.230 ± 0.033	1.213 ± 0.128	70.056 ± 10.781	0.329 ± 0.095	0.374 ± 0.067
	random	0.817 ± 0.061	0.267 ± 0.038	**1.059 ± 0.106**	87.486 ± 15.035	0.361 ± 0.084	0.399 ± 0.078
	approx.	0.817 ± 0.050	0.223 ± 0.034	1.237 ± 0.128	66.452 ± 10.478	0.328 ± 0.094	0.364 ± 0.069

D.8.2 Stability with respect to the edge contraction sequence

Because computing multiset of diversity curves over all possible edge contractions is infeasible in practice, particularly on large graphs, we have to limit ourselves to evaluating a smaller selection of possible edge contraction sequences. This will introduce some randomness in the selection of the edge contraction sequence. Even when choosing the edge contraction sequence by using a process that is based only on the graph structure, edges in symmetric graphs that cannot structurally be distinguished will obtain the same importance score. Only a canonical labelling of the edges in the graph would give us a unique score for each edge that is invariant under isomorphisms, but this difficult problem related to the graph isomorphism problem is infeasible to use in practice [1]. Hence, it is desirable to investigate the stability of diversity curves with respect to the randomness of choosing a subset of all possible edge contractions. In this ablation we study how stable the diversity curves are with respect to different edge contraction sequences and random seeds.

We follow the experiment from Section 4.2 to distinguish between the different random graph distributions and choose the same parameter settings as described in Section D.2. For each graph we compute its diversity curve based only on one sequence of random edge contractions and repeat the computation over 10 different random seeds. As a first measure of the stability, we consider the distribution of the mean accuracies from the kNN-classification over the 10 different seeds. We find that the average and standard deviation of the mean accuracies obtained directly from the embedding through the diversity curves is 0.88 ± 0.009.

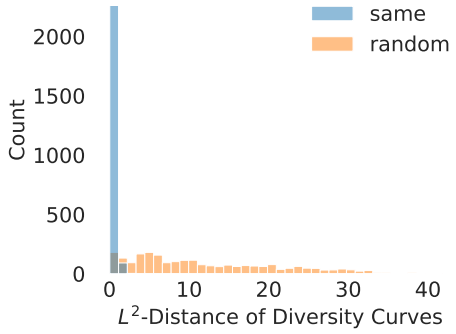

Figure S.11: Distribution of differences between the diversity curve of each graph in each seed and its mean diversity curve over all seeds (*same*) or when comparing to a randomly chosen graph from another graph model (*random*).

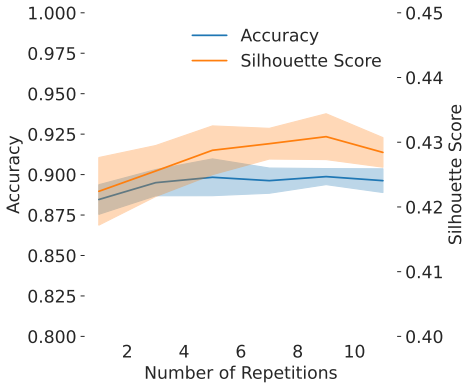

Figure S.12: Mean accuracy and silhouette score (mean±std) at distinguishing graph models over 10 different seeds for varying number of repetitions over which we compute and average the diversity curves.

Secondly, we compare the differences between the diversity curves themselves. To do this, we compute the mean diversity curve of each graph over all 10 seeds and then for each graph in each seed we once get the L^2-distance to the mean curve of the same graph and to a randomly selected graph from a different graph distribution. The mean and standard deviation for the distance distributions of the diversity curve between the same graphs is 0.48 ± 0.33, while for the distance distribution of the diversity curve with a randomly chosen graph from a different distribution is 14.68 ± 8.99. Figure S.11 shows a histogram of the two distributions. To conclude, we find that empirically the diversity curves are stable with respect to different random edge contractions, validating our approach of considering a subset of the multiset of the diversity curves for our empirical evaluation.

As we saw, the choice of random edge contraction sequence introduces some variance in the diversity curves, thus in the experiments we choose to repeat the diversity curves over 3 different random edge contraction sequences and take the mean. This ablation addresses the influence on the number of repeats in computing the diversity curves. Concretely, we repeat the experiment on the simulated graph dataset for distinguishing between different distributions from Table 1 while varying the number of repetitions for which we compute diversity curves, ranging over $\{1, 3, 5, 7, 9, 11\}$. For each fixed number of repetitions, we repeat the experiment over 10 random seeds and report the mean and standard deviation over the different seeds of the mean accuracy and mean silhouette score in Figure S.12. We find that for only one repetition, the scores are lowest but that overall the diversity curves remain stable with respect to varying the number of repetitions.

D.8.3 Choice of evaluation scales, expressivity, and computational cost

In this ablation study, we consider the trade-off between the choice of evaluation scales, the computational efficiency and expressivity. We repeat the experiment on distinguishing simulated graph distributions from Table 1 by varying the numbers of evaluation scales used for computing diversity curves, evenly chosen between 1 and 29. The results are reported in Table 6. Computing the diversity curves at fewer scales reduces the runtimes while the accuracy remains reasonably high, only decreasing for lower numbers of scales. This demonstrates that diversity curves remain robust and expressive across scales.

D.8.4 Performance across up- or downsampling

To investigate the impact of the up- or downsampling strategies, we modify the experiment on distinguishing parameter choices on simulated graphs from Table 2 to consider down- or up-sampling operations individually. We compute coarsened graph representations that only use downsampling to less than 10 nodes, and another version where we upsample every graph to between 31 and 50 nodes. The results are found in Table S.16. In this setup, exclusively downsampling reaches an accuracy below the main results reported in our paper, likely due to the low cardinalities, while upsampling alone shows an accuracy comparable to the accuracies reported in Table 2. Overall, the strength of

Table S.16: Downsample and upsample accuracy across graph models.

	ER	RG	RP	SBM
Downsample Accuracy	0.910 ± 0.062	0.900 ± 0.060	0.906 ± 0.090	0.800 ± 0.100
Upsample Accuracy	**1.000 ± 0.000**	**0.978 ± 0.037**	**0.983 ± 0.025**	**0.856 ± 0.067**

our methods lies in being able to upsample some graphs and downsample others to the same node sizes to compare them across a shared range of cardinalities.

D.9 Extended results on graph classification datasets

Note that our work focuses on evaluating the utility of structural descriptors for unsupervised tasks across coarsening levels. Our methods are designed to encode geometric and structural properties of unattributed graphs and meshes. When applied to attributed graphs and graph classification tasks, for example from TU datasets [59], we do not expect unsupervised methods to reach state-of-the-art performance, but we can nevertheless compare the performance of different unsupervised baselines.

Hence, we provide extended results on MCF-7 and MOLT-4 in Tables S.17 and S.18, two large molecular graph datasets from TU datasets [59], containing 27,770 and 39,765 graphs respectively. Specifically, we compare kNN classification accuracies across 10-fold stratified cross-validation following the experimental setup reported in our paper. Some of the kernel methods were not evaluated due to computational bottlenecks and memory allocation limitations. This highlights the computational advantages of our approach for processing large graph datasets over less scalable baseline methods.

Overall, diversity curves reach competitive performance and comparable accuracies to relevant baseline methods, such as FEATHER, SpectralZoo, TopER, and kernel methods. To interpret the results, we note that it remains uncertain whether the suggested graph datasets show more meaningful local graph structures or global differences in graph structure, which could warrant further investigation. Nevertheless, our methods show satisfactory performance across these graph classification tasks.

Table S.17: Accuracy on MCF-7 and MOLT-4.

Method	MCF-7	MOLT-4
Diversity Curves (Diffusion Dist.)	0.915 ± 0.002	0.919 ± 0.002
Diversity Curves (Shortest Path)	0.916 ± 0.002	0.919 ± 0.001
Graph2Vec	**0.923 ± 0.002**	**0.927 ± 0.001**
NetLSD	0.919 ± 0.003	0.922 ± 0.001
FEATHER	0.915 ± 0.002	0.919 ± 0.002
Spectral Zoo	0.916 ± 0.001	0.920 ± 0.000
Kernel Core Framework	0.917 ± 0.004	OOM
Kernel Shortest Path	0.917 ± 0.001	OOM
TopER Degree	0.915 ± 0.002	0.919 ± 0.003
TopER Forman-Ricci	0.914 ± 0.002	0.920 ± 0.002
TopER Popularity	0.914 ± 0.002	0.918 ± 0.001

Table S.18: AUROC on MCF-7 and MOLT-4.

Method	MCF-7	MOLT-4
Diversity Curves (Diffusion Dist.)	0.668 ± 0.007	0.610 ± 0.012
Diversity Curves (Shortest Path)	0.650 ± 0.018	0.619 ± 0.021
Graph2Vec	0.692 ± 0.013	**0.699 ± 0.013**
NetLSD	**0.698 ± 0.018**	0.684 ± 0.017
FEATHER	0.669 ± 0.017	0.663 ± 0.023
Spectral Zoo	0.525 ± 0.009	0.510 ± 0.012
Kernel Core Framework	**0.698 ± 0.022**	OOM
Kernel Shortest Path	0.657 ± 0.011	OOM
TopER Degree	0.632 ± 0.019	0.612 ± 0.008
TopER Forman-Ricci	0.631 ± 0.017	0.617 ± 0.014
TopER Popularity	0.638 ± 0.025	0.628 ± 0.014

NeurIPS Paper Checklist

1. **Claims**

   Question: Do the main claims made in the abstract and introduction accurately reflect the paper's contributions and scope?

   Answer: [Yes]

   Justification: The claims and contributions described in our paper are all supported either by theoretical or empirical evidence on the performance of our methods across a range of graph-level tasks taking into account baseline methods and existing literature on graph representation learning.

   Guidelines:

   - The answer [N/A] means that the abstract and introduction do not include the claims made in the paper.
   - The abstract and/or introduction should clearly state the claims made, including the contributions made in the paper and important assumptions and limitations. A [No] or [N/A] answer to this question will not be perceived well by the reviewers.
   - The claims made should match theoretical and experimental results, and reflect how much the results can be expected to generalize to other settings.
   - It is fine to include aspirational goals as motivation as long as it is clear that these goals are not attained by the paper.

2. **Limitations**

   Question: Does the paper discuss the limitations of the work performed by the authors?

   Answer: [Yes]

   Justification: We include a section on the practical and theoretical limitations of our methods in Section 5. We discuss computational efficiency in Section 3.2 and Section D.7. Further, we reflect upon the relevant practical and theoretical imitations when detailing each experiment and theoretical result in the appendix and main text.

   Guidelines:

   - The answer [N/A] means that the paper has no limitation while the answer [No] means that the paper has limitations, but those are not discussed in the paper.
   - The authors are encouraged to create a separate "Limitations" section in their paper.
   - The paper should point out any strong assumptions and how robust the results are to violations of these assumptions (e.g., independence assumptions, noiseless settings, model well-specification, asymptotic approximations only holding locally). The authors should reflect on how these assumptions might be violated in practice and what the implications would be.
   - The authors should reflect on the scope of the claims made, e.g., if the approach was only tested on a few datasets or with a few runs. In general, empirical results often depend on implicit assumptions, which should be articulated.
   - The authors should reflect on the factors that influence the performance of the approach. For example, a facial recognition algorithm may perform poorly when image resolution is low or images are taken in low lighting. Or a speech-to-text system might not be used reliably to provide closed captions for online lectures because it fails to handle technical jargon.
   - The authors should discuss the computational efficiency of the proposed algorithms and how they scale with dataset size.
   - If applicable, the authors should discuss possible limitations of their approach to address problems of privacy and fairness.
   - While the authors might fear that complete honesty about limitations might be used by reviewers as grounds for rejection, a worse outcome might be that reviewers discover limitations that aren't acknowledged in the paper. The authors should use their best judgment and recognize that individual actions in favor of transparency play an important role in developing norms that preserve the integrity of the community. Reviewers will be specifically instructed to not penalize honesty concerning limitations.

3. **Theory assumptions and proofs**

   Question: For each theoretical result, does the paper provide the full set of assumptions and a complete (and correct) proof?

   Answer: [Yes]

   Justification: We report all theoretical results in detail in Section C and a condensed summary in Section 3.2. All theorems are formally stated and all relevant assumptions are detailed throughout our theoretical analysis.

   Guidelines:

   - The answer [N/A] means that the paper does not include theoretical results.
   - All the theorems, formulas, and proofs in the paper should be numbered and cross-referenced.
   - All assumptions should be clearly stated or referenced in the statement of any theorems.
   - The proofs can either appear in the main paper or the supplemental material, but if they appear in the supplemental material, the authors are encouraged to provide a short proof sketch to provide intuition.
   - Inversely, any informal proof provided in the core of the paper should be complemented by formal proofs provided in appendix or supplemental material.
   - Theorems and Lemmas that the proof relies upon should be properly referenced.

4. **Experimental result reproducibility**

   Question: Does the paper fully disclose all the information needed to reproduce the main experimental results of the paper to the extent that it affects the main claims and/or conclusions of the paper (regardless of whether the code and data are provided or not)?

   Answer: [Yes]

   Justification: We provide extended experimental details for each experiment in Section D as well as in Section 4, which can be used to reproduce our main results. All existing datasets and codebases which our results are based on are referenced throughout the paper, enabling reproducibility. We publish the code for reproducing all of our results on GitHub. As our main contribution is a novel graph representation method, our algorithm in described in detail throughout the main text in Section 3.1 and in the extended methods description in Section B.1.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - If the paper includes experiments, a [No] answer to this question will not be perceived well by the reviewers: Making the paper reproducible is important, regardless of whether the code and data are provided or not.
   - If the contribution is a dataset and/or model, the authors should describe the steps taken to make their results reproducible or verifiable.
   - Depending on the contribution, reproducibility can be accomplished in various ways. For example, if the contribution is a novel architecture, describing the architecture fully might suffice, or if the contribution is a specific model and empirical evaluation, it may be necessary to either make it possible for others to replicate the model with the same dataset, or provide access to the model. In general. releasing code and data is often one good way to accomplish this, but reproducibility can also be provided via detailed instructions for how to replicate the results, access to a hosted model (e.g., in the case of a large language model), releasing of a model checkpoint, or other means that are appropriate to the research performed.
   - While NeurIPS does not require releasing code, the conference does require all submissions to provide some reasonable avenue for reproducibility, which may depend on the nature of the contribution. For example
     (a) If the contribution is primarily a new algorithm, the paper should make it clear how to reproduce that algorithm.
     (b) If the contribution is primarily a new model architecture, the paper should describe the architecture clearly and fully.

(c) If the contribution is a new model (e.g., a large language model), then there should either be a way to access this model for reproducing the results or a way to reproduce the model (e.g., with an open-source dataset or instructions for how to construct the dataset).

(d) We recognize that reproducibility may be tricky in some cases, in which case authors are welcome to describe the particular way they provide for reproducibility. In the case of closed-source models, it may be that access to the model is limited in some way (e.g., to registered users), but it should be possible for other researchers to have some path to reproducing or verifying the results.

5. **Open access to data and code**

   Question: Does the paper provide open access to the data and code, with sufficient instructions to faithfully reproduce the main experimental results, as described in supplemental material?

   Answer: [Yes]

   Justification: We provide a link to our GitHub for fully reproducing our methods and all results. We provide clear instructions on how to run our algorithm and the experiments.

   Guidelines:

   - The answer [N/A] means that paper does not include experiments requiring code.
   - Please see the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - While we encourage the release of code and data, we understand that this might not be possible, so [No] is an acceptable answer. Papers cannot be rejected simply for not including code, unless this is central to the contribution (e.g., for a new open-source benchmark).
   - The instructions should contain the exact command and environment needed to run to reproduce the results. See the NeurIPS code and data submission guidelines (https://neurips.cc/public/guides/CodeSubmissionPolicy) for more details.
   - The authors should provide instructions on data access and preparation, including how to access the raw data, preprocessed data, intermediate data, and generated data, etc.
   - The authors should provide scripts to reproduce all experimental results for the new proposed method and baselines. If only a subset of experiments are reproducible, they should state which ones are omitted from the script and why.
   - At submission time, to preserve anonymity, the authors should release anonymized versions (if applicable).
   - Providing as much information as possible in supplemental material (appended to the paper) is recommended, but including URLs to data and code is permitted.

6. **Experimental setting/details**

   Question: Does the paper specify all the training and test details (e.g., data splits, hyperparameters, how they were chosen, type of optimizer) necessary to understand the results?

   Answer: [Yes]

   Justification: We clarify the choice of classifier, embedding model, and data splits throughout our paper and in the extended experimental description in Section D. Altogether, this provides the necessary detail to reproduce our main results.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The experimental setting should be presented in the core of the paper to a level of detail that is necessary to appreciate the results and make sense of them.
   - The full details can be provided either with the code, in appendix, or as supplemental material.

7. **Experiment statistical significance**

   Question: Does the paper report error bars suitably and correctly defined or other appropriate information about the statistical significance of the experiments?

Answer: [Yes]

Justification: When evaluating model performance, we report standard deviations across all of our experiments, and take care to interpret the performance of our method accordingly. Each experimental section and the extended descriptions in Section D specify whether these deviations were reported across random seed, cross validation splits, or resamples of the data. Further, we use diversity curves for hypothesis testing in Section 4.4 showing that our methods themselves can be used for statistical analysis.

Guidelines:

- The answer [N/A] means that the paper does not include experiments.
- The authors should answer [Yes] if the results are accompanied by error bars, confidence intervals, or statistical significance tests, at least for the experiments that support the main claims of the paper.
- The factors of variability that the error bars are capturing should be clearly stated (for example, train/test split, initialization, random drawing of some parameter, or overall run with given experimental conditions).
- The method for calculating the error bars should be explained (closed form formula, call to a library function, bootstrap, etc.)
- The assumptions made should be given (e.g., Normally distributed errors).
- It should be clear whether the error bar is the standard deviation or the standard error of the mean.
- It is OK to report 1-sigma error bars, but one should state it. The authors should preferably report a 2-sigma error bar than state that they have a 96% CI, if the hypothesis of Normality of errors is not verified.
- For asymmetric distributions, the authors should be careful not to show in tables or figures symmetric error bars that would yield results that are out of range (e.g., negative error rates).
- If error bars are reported in tables or plots, the authors should explain in the text how they were calculated and reference the corresponding figures or tables in the text.

8. **Experiments compute resources**

   Question: For each experiment, does the paper provide sufficient information on the computer resources (type of compute workers, memory, time of execution) needed to reproduce the experiments?

   Answer: [Yes]

   Justification: We specify the maximal compute resources used to run our experiments in Section B.2.

   Guidelines:

   - The answer [N/A] means that the paper does not include experiments.
   - The paper should indicate the type of compute workers CPU or GPU, internal cluster, or cloud provider, including relevant memory and storage.
   - The paper should provide the amount of compute required for each of the individual experimental runs as well as estimate the total compute.
   - The paper should disclose whether the full research project required more compute than the experiments reported in the paper (e.g., preliminary or failed experiments that didn't make it into the paper).

9. **Code of ethics**

   Question: Does the research conducted in the paper conform, in every respect, with the NeurIPS Code of Ethics https://neurips.cc/public/EthicsGuidelines?

   Answer: [Yes]

   Justification: We have reviewed and comply with the NeurIPS Code of Ethics.

   Guidelines:

   - The answer [N/A] means that the authors have not reviewed the NeurIPS Code of Ethics.

- If the authors answer [No], they should explain the special circumstances that require a deviation from the Code of Ethics.
- The authors should make sure to preserve anonymity (e.g., if there is a special consideration due to laws or regulations in their jurisdiction).

10. **Broader impacts**

    Question: Does the paper discuss both potential positive societal impacts and negative societal impacts of the work performed?

    Answer: [N/A]

    Justification: We do consider the contributions of our work to benefit foundational machine research as a whole, hence there are many potential consequences. However, we do not find any specific harmful society implications or immediate negative consequences of our work that we feel must be highlighted here.

    Guidelines:

    - The answer [N/A] means that there is no societal impact of the work performed.
    - If the authors answer [N/A] or [No], they should explain why their work has no societal impact or why the paper does not address societal impact.
    - Examples of negative societal impacts include potential malicious or unintended uses (e.g., disinformation, generating fake profiles, surveillance), fairness considerations (e.g., deployment of technologies that could make decisions that unfairly impact specific groups), privacy considerations, and security considerations.
    - The conference expects that many papers will be foundational research and not tied to particular applications, let alone deployments. However, if there is a direct path to any negative applications, the authors should point it out. For example, it is legitimate to point out that an improvement in the quality of generative models could be used to generate Deepfakes for disinformation. On the other hand, it is not needed to point out that a generic algorithm for optimizing neural networks could enable people to train models that generate Deepfakes faster.
    - The authors should consider possible harms that could arise when the technology is being used as intended and functioning correctly, harms that could arise when the technology is being used as intended but gives incorrect results, and harms following from (intentional or unintentional) misuse of the technology.
    - If there are negative societal impacts, the authors could also discuss possible mitigation strategies (e.g., gated release of models, providing defenses in addition to attacks, mechanisms for monitoring misuse, mechanisms to monitor how a system learns from feedback over time, improving the efficiency and accessibility of ML).

11. **Safeguards**

    Question: Does the paper describe safeguards that have been put in place for responsible release of data or models that have a high risk for misuse (e.g., pre-trained language models, image generators, or scraped datasets)?

    Answer: [N/A]

    Justification: We do not release any data or models that have a high risk of misuse. Our main contribution is the publication of method for learning graph-level representations, which we feel does not pose a high safety risk.

    Guidelines:

    - The answer [N/A] means that the paper poses no such risks.
    - Released models that have a high risk for misuse or dual-use should be released with necessary safeguards to allow for controlled use of the model, for example by requiring that users adhere to usage guidelines or restrictions to access the model or implementing safety filters.
    - Datasets that have been scraped from the Internet could pose safety risks. The authors should describe how they avoided releasing unsafe images.
    - We recognize that providing effective safeguards is challenging, and many papers do not require this, but we encourage authors to take this into account and make a best faith effort.

12. **Licenses for existing assets**

    Question: Are the creators or original owners of assets (e.g., code, data, models), used in the paper, properly credited and are the license and terms of use explicitly mentioned and properly respected?

    Answer: [Yes]

    Justification: We cite existing assets and publications our work is based on throughout our paper and throughout the extended experimental description in Section D.

    Guidelines:

    - The answer [N/A] means that the paper does not use existing assets.
    - The authors should cite the original paper that produced the code package or dataset.
    - The authors should state which version of the asset is used and, if possible, include a URL.
    - The name of the license (e.g., CC-BY 4.0) should be included for each asset.
    - For scraped data from a particular source (e.g., website), the copyright and terms of service of that source should be provided.
    - If assets are released, the license, copyright information, and terms of use in the package should be provided. For popular datasets, `paperswithcode.com/datasets` has curated licenses for some datasets. Their licensing guide can help determine the license of a dataset.
    - For existing datasets that are re-packaged, both the original license and the license of the derived asset (if it has changed) should be provided.
    - If this information is not available online, the authors are encouraged to reach out to the asset's creators.

13. **New assets**

    Question: Are new assets introduced in the paper well documented and is the documentation provided alongside the assets?

    Answer: [Yes]

    Justification: The main new asset introduced in our publication is an algorithm for computing diversity curves, which are novel graph representations. We detail this algorithm throughout our paper and publish code to use our methods with the publication of this manuscript.

    Guidelines:

    - The answer [N/A] means that the paper does not release new assets.
    - Researchers should communicate the details of the dataset/code/model as part of their submissions via structured templates. This includes details about training, license, limitations, etc.
    - The paper should discuss whether and how consent was obtained from people whose asset is used.
    - At submission time, remember to anonymize your assets (if applicable). You can either create an anonymized URL or include an anonymized zip file.

14. **Crowdsourcing and research with human subjects**

    Question: For crowdsourcing experiments and research with human subjects, does the paper include the full text of instructions given to participants and screenshots, if applicable, as well as details about compensation (if any)?

    Answer: [N/A]

    Justification: This work does not include any research or crowdsourcing with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Including this information in the supplemental material is fine, but if the main contribution of the paper involves human subjects, then as much detail as possible should be included in the main paper.

- According to the NeurIPS Code of Ethics, workers involved in data collection, curation, or other labor should be paid at least the minimum wage in the country of the data collector.

15. **Institutional review board (IRB) approvals or equivalent for research with human subjects**

    Question: Does the paper describe potential risks incurred by study participants, whether such risks were disclosed to the subjects, and whether Institutional Review Board (IRB) approvals (or an equivalent approval/review based on the requirements of your country or institution) were obtained?

    Answer: [N/A]

    Justification: This work does not include any research or crowdsourcing with human subjects.

    Guidelines:

    - The answer [N/A] means that the paper does not involve crowdsourcing nor research with human subjects.
    - Depending on the country in which research is conducted, IRB approval (or equivalent) may be required for any human subjects research. If you obtained IRB approval, you should clearly state this in the paper.
    - We recognize that the procedures for this may vary significantly between institutions and locations, and we expect authors to adhere to the NeurIPS Code of Ethics and the guidelines for their institution.
    - For initial submissions, do not include any information that would break anonymity (if applicable), such as the institution conducting the review.

16. **Declaration of LLM usage**

    Question: Does the paper describe the usage of LLMs if it is an important, original, or non-standard component of the core methods in this research? Note that if the LLM is used only for writing, editing, or formatting purposes and does *not* impact the core methodology, scientific rigor, or originality of the research, declaration is not required.

    Answer: [N/A]

    Justification: The core method development in this work does not involve LLMs as any important, original, or non-standard components.

    Guidelines:

    - The answer [N/A] means that the core method development in this research does not involve LLMs as any important, original, or non-standard components.
    - Please refer to our LLM policy in the NeurIPS handbook for what should or should not be described.