Deep Graph Mapper: Seeing Graphs through the Neural Lens

Cristian Bodnar*, Cătălina Cangea*, Pietro Liò
Department of Computer Science and Technology
University of Cambridge

Abstract

Graph summarisation has received much attention lately, with various works tackling the challenge of defining pooling operators on data regions with arbitrary structures. These contrast the grid-like ones encountered in image inputs, where techniques such as max-pooling have been enough to show empirical success. In this work[1], we merge the Mapper algorithm with the expressive power of graph neural networks to produce topologically-grounded graph summaries. We demonstrate the suitability of Mapper as a topological framework for graph pooling by proving that Mapper is a generalisation of pooling methods based on soft cluster assignments. Building upon this, we show how easy it is to design novel pooling algorithms that obtain competitive results with other state-of-the-art methods.

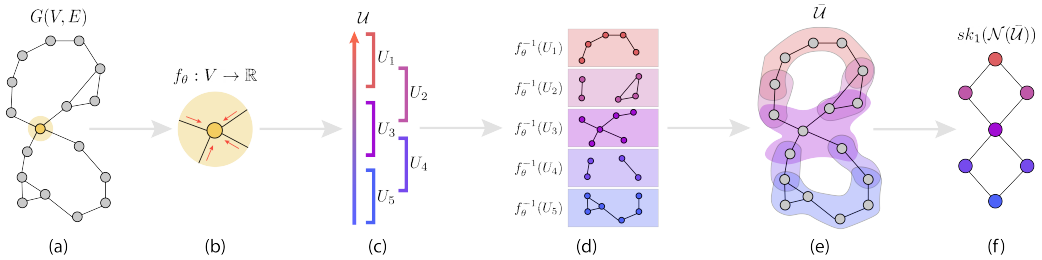


Figure 1: An illustration of the Deep Graph Mapper (DGM) algorithm where, for simplicity, a graph neural network (GNN) approximates a 'height' function over the nodes in the plane of the diagram. The input graph (a) is passed through the GNN, which maps the vertices of the graph to a real number (the height) (b–c). Given a cover $\mathcal{U}$ of the image of the GNN (c), the refined pull back cover $\bar{\mathcal{U}}$ is computed (d–e). The 1-skeleton of the nerve of the pull back cover provides the visual summary of the graph (f). The diagram is inspired by Hajij et al. [2018].

1 Introduction

We introduce Deep Graph Mapper (DGM), a synthesis of Mapper [Singh et al., 2007], an algorithm from the field of Topological Data Analysis (TDA) [Chazal and Michel, 2017], and graph neural networks (GNNs) [Scarselli et al., 2008, Battaglia et al., 2018, Bronstein et al., 2017]. We further demonstrate an important connection between Mapper visualisations and graph pooling methods. Namely, we prove that Mapper is a generalisation of pooling methods based on soft cluster assignments, which include state-of-the-art algorithms like minCUT [Bianchi et al., 2019] and DiffPool [Ying et al., 2018]. Building upon this topological perspective of graph pooling, we propose

[1]Code to reproduce models and experimental results is available at https://github.com/crisbodnar/dgm.

Topological Data Analysis and Beyond Workshop at the 34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

two algorithms based on Mapper. These leverage fully-differentiable and fixed PageRank-based lens functions, respectively, achieving results competitive with other state-of-the-art pooling methods on graph classification benchmarks.

2 Related work

Graph pooling algorithms have already been considerably explored within GNN frameworks for graph classification. Luzhnica et al. [2019b] propose a topological approach to pooling, which coarsens the graph by aggregating its maximal cliques into new clusters. However, cliques are local topological features, whereas our methods leverage a global perspective of the graph during pooling. Two paradigms distinguish themselves among learnable pooling layers: Top-k pooling based on a learnable ranking [Gao and Ji, 2019], and learning the cluster assignment [Ying et al., 2018] with additional entropy and link prediction losses for more stable training (DiffPool). Following these two trends, several variants and incremental improvements have been proposed. The Top-k approach is explored in conjunction with jumping-knowledge networks [Cangea et al., 2018], attention [Lee et al., 2019, Huang et al., 2019] and self-attention for cluster assignment [Ranjan et al., 2019]. Similarly to DiffPool, the method suggested by Bianchi et al. [2019] uses several loss terms to enforce clusters with strongly connected nodes, similar sizes and orthogonal assignments. A different approach is also proposed by Ma et al. [2019], who leverage spectral clustering.

3 Background

In this section, we briefly review Mapper [Singh et al., 2007], with a focus on graph domains [Hajij et al., 2018].

Definition 3.1. Let X be a topological space, $f : X \to \mathbb{R}^d, d \geq 1$ a continuous function, and $\mathcal{U} = (U_i)_{i\in I}$ a cover of $\mathbb{R}^d$. Then, the **pull back cover** $f^*(\mathcal{U})$ of X induced by $(f,\mathcal{U})$ is the collection of open sets $f^{-1}(U_i), i \in I$, for some indexing set I. The cover of X formed by the connected components of all preimages in the pull back cover $f^*(\mathcal{U})$ is called the **refined pull back cover**, which we denote by $\bar{\mathcal{U}} = (\bar{U}_j)_{j\in J}$, where J is an indexing set.

Definition 3.2. Let X be a topological space with an open cover $\mathcal{U} = (U_i)_{i\in I}$. The **1-skeleton of the nerve** $\mathcal{N}(\mathcal{U})$ of $\mathcal{U}$, which we denote by $sk_1(\mathcal{N}(\mathcal{U}))$, is the graph with vertices given by $(v_i)_{i\in I}$, where two vertices v_i, v_j are connected if and only if $U_i \cap U_j \neq \emptyset$.

Given a graph $G = (V, E)$, a carefully chosen **lens function** $f : V \to \mathbb{R}^d$ and cover $\mathcal{U}$ of $\mathbb{R}^d$, Mapper first computes the associated pull back cover $f^*(\mathcal{U})$. Using a clustering algorithm of choice, it then clusters each of the sets of vertices $f^{-1}(U_i)$ in $f^*(\mathcal{U})$ to obtain the refined pull back cover. Finally, Mapper produces a summarised graph visualisation by computing the 1-skeleton of the nerve of the refined pull back cover, $sk_1(\mathcal{N}(\bar{\mathcal{U}}))$. Informally, the soft clusters formed by the refined pull back become the nodes of this graph, with nodes corresponding to overlapping clusters being connected by an edge.

4 Mapper for pooling

4.1 Mapper and soft cluster assignments

An early suggestion that Mapper could be suitable for graph pooling is given by the fact that it constitutes a generalisation of binary spectral clustering, as observed by Hajij et al. [2018] (proof in Appendix F). Here, we provide a much stronger result in that direction, showing that Mapper is a generalisation of all pooling methods based on soft-cluster assignments.

Let $G(V, E)$ be a graph with self-loops for each node and adjacency matrix $\mathbf{A}$. *Soft cluster assignment pooling methods* use a soft cluster assignment matrix $\mathbf{S} \in \mathbb{R}^{N\times K}$, where N is the number of nodes in the graph and K is the number of clusters, and compute the new adjacency matrix of the graph via $\mathbf{A}' = \mathbf{S}^T\mathbf{AS}$. Equivalently, two soft clusters become connected if and only if there is a common edge between them.

Proposition 4.1. There exists a graph $G'(V', E')$ derived from G, a soft cluster assignment $\mathbf{S}'$ of G' based on $\mathbf{S}$, and a cover $\mathcal{U}$ of the $(K-1)$-dimensional unit simplex $\triangle_{K-1}$, such that the 1-skeleton of the nerve of the pull back cover induced by $(\mathbf{S}', \mathcal{U})$ is isomorphic with the graph defined by $\mathbf{A}'$.

A detailed proof and its diagrammatic version can be found in Appendix C. We hope that this result will enable theoreticians to study pooling operators through the topological and statistical properties of Mapper [Carriere et al., 2018, Carrière and Oudot, 2018, Dey et al., 2017]. At the same time, we encourage practitioners to take advantage of it and design new pooling methods in terms of a well-chosen lens function f and cover $\mathcal{U}$ for its image. To illustrate this idea and showcase the benefits of this new perspective over graph pooling methods, we introduce two Mapper-based operators.

4.2 Differentiable Mapper pooling (DMP)

In traditional applications of Mapper, the lens function and pullback cover are fixed—here, we address the challenge of learning both via backpropagating through the pull back computation. We start by replacing the cover of n overlapping intervals over the real line, described in the previous section, with a cover formed of overlapping RBF kernels $\phi(x, x_i) = \exp(-\frac{\|x-x_i\|^2}{\delta})$, evaluated at n fixed locations x_i. The overlap between these kernels can be adjusted through the scale δ of the kernels. The soft cluster assignment matrix $\mathbf{S}$ is given by the normalised kernel values:

$$S_{ij} = \frac{\phi(\sigma(f_\theta(\mathbf{X}_l))_i, x_j)}{\sum_{j=1}^{n} \phi(\sigma(f_\theta(\mathbf{X}_l))_i, x_j)}, \tag{1}$$

where the lens function f_θ is a GNN layer, σ is a sigmoid function ensuring the outputs are in $[0, 1]$, and $\mathbf{X}_l$ are the node features at layer l. The more closely a node is mapped to a location x_i, the more it belongs to cluster i. By Proposition 4.1, we can compute the adjacency matrix of the pooled graph as $\mathbf{S}^T\mathbf{AS}$; the features are given by $\mathbf{S}^T\mathbf{X}$. This method can be considered a version of DiffPool [Ying et al., 2018], where the low-entropy constraint on the cluster assignment distribution is topologically satisfied, since a point cannot be equally close to many other points on a line. Therefore, each node will belong only to a few clusters if the scale δ is appropriately set. This is illustrated in Appendix G.

4.3 Mapper-based PageRank (MPR) pooling

To evaluate the effectiveness of the differentiable pooling operator, we also consider a fixed and scalable non-differentiable lens function $f : V \to \mathbb{R}$ that is given by the normalised PageRank (PR) [Page et al., 1999] of the nodes. The PageRank function assigns an importance value to each of the nodes based on their connectivity, according to the well-known recurrence relation $f(\mathbf{X})_i \triangleq \mathbf{PR}_i = \sum_{j \in N(i)} \frac{\mathbf{PR}_j}{|N(i)|}$, where $N(i)$ represents the set of neighbours of the i-th node in the graph. The resulting scores are values in $[0, 1]$ which reflect the probability of a random walk through the graph to end in a given node. Using the previously described overlapping intervals cover $\mathcal{U}$, the pull back cover elements form a soft cluster assignment matrix $\mathbf{S}$:

$$S_{ij} = \frac{\mathbb{I}_{i \in f^{-1}(U_j)}}{|\{U_k | i \in f^{-1}(U_k)\}|} \tag{2}$$

where U_n is the n-th overlapping interval in the cover $\mathcal{U}$ of $[0, 1]$. It can be observed that the resulting clusters contain nodes with similar PageRank scores. Intuitively, this pooling method exploits the power-law distributions typically encountered in social data scenarios—it merges the (usually few) highly connected nodes in the graph, while at the same time clustering the (typically many) dangling nodes that have a normalised PageRank score closer to zero. Therefore, this method favours the information attached to the most 'important' nodes of the graph. The adjacency matrix of the pooled graph and the features are computed in the same manner as for DMP.

4.4 Model

For the graph classification task, each example G is represented by a tuple $(\mathbf{X}, \mathbf{A})$, where $\mathbf{X}$ is the node feature matrix and $\mathbf{A}$ is the adjacency matrix. Both our graph embedding and classification networks consist of a sequence of graph convolutional layers [Kipf and Welling, 2016]; the l-th layer operates on its input feature matrix as follows:

$$\mathbf{X}_{l+1} = \sigma(\hat{\mathbf{D}}^{-\frac{1}{2}}\hat{\mathbf{A}}\hat{\mathbf{D}}^{-\frac{1}{2}}\mathbf{X}_l\mathbf{W}_l), \tag{3}$$

where $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{I}$ is the adjacency matrix with self-loops, $\hat{\mathbf{D}}$ is the normalised node degree matrix and σ is the activation function. After E layers, the embedding network simply outputs node features

Table 1: Results obtained on classification benchmarks. Accuracy measures with 95% confidence intervals are reported. The highest result is **bolded** and the second highest is underlined. The first four columns correspond to molecular graphs, while the others, to social graphs. Our models perform competitively with other state-of-the-art methods.

Model	D&D	Mutag	NCI1	Proteins	Collab	IMDB-B	IMDB-M	Reddit-B	Reddit-5k
DMP (Ours)	77.3 ± 3.6	84.0 ± 8.6	70.4 ± 4.2	$\mathbf{75.3 \pm 3.3}$	81.4 ± 1.2	$\mathbf{73.8 \pm 4.5}$	$\mathbf{50.9 \pm 2.5}$	86.2 ± 6.8	51.9 ± 2.1
MPR (Ours)	$\mathbf{78.2 \pm 3.4}$	80.3 ± 6.0	69.8 ± 1.8	75.2 ± 2.2	$\mathbf{81.5 \pm 1.0}$	73.4 ± 2.7	50.6 ± 2.0	86.3 ± 4.8	52.3 ± 1.6
Top-k	75.1 ± 2.2	82.5 ± 6.8	67.9 ± 2.3	74.8 ± 3.0	75.0 ± 1.1	69.6 ± 3.8	45.0 ± 2.8	79.4 ± 7.4	48.5 ± 1.1
minCUT	77.6 ± 3.1	82.9 ± 6.0	68.8 ± 2.1	73.5 ± 2.9	79.9 ± 0.8	70.7 ± 3.5	50.6 ± 2.1	$\mathbf{87.2 \pm 5.0}$	$\mathbf{52.9 \pm 1.3}$
DiffPool	77.9 ± 2.4	$\mathbf{94.7 \pm 7.1}$	68.1 ± 2.1	74.2 ± 0.3	81.3 ± 0.1	72.4 ± 3.1	50.3 ± 1.8	79.0 ± 1.1	50.4 ± 1.7
WL	77.4 ± 2.6	74.5 ± 6.5	$\mathbf{76.4 \pm 2.7}$	74.7 ± 3.2	78.5 ± 1.1	72.1 ± 3.1	50.7 ± 2.9	66.7 ± 10.4	49.2 ± 1.4
Flat	69.9 ± 2.2	71.8 ± 4.3	65.5 ± 1.7	70.2 ± 2.6	80.9 ± 1.4	73.6 ± 4.2	48.5 ± 2.4	70.0 ± 10.8	49.5 ± 1.7
avg-MLP	63.7 ± 1.4	69.1 ± 5.8	55.7 ± 2.8	61.8 ± 1.7	74.8 ± 1.3	71.5 ± 2.9	49.5 ± 2.2	53.6 ± 6.2	45.9 ± 1.6

$\mathbf{X}_{L_E}$, which are subsequently processed by a pooling layer to coarsen the graph. The classification network first takes as input node features of the Mapper-pooled graph[2] $\mathbf{X}_{\text{MG}}$ (as described in Figure 1) and passes them through L_C graph convolutional layers. Following this, the network computes a graph summary given by the feature-wise node average and applies a final linear layer which predicts the class, $y = \text{softmax}\big(\frac{1}{|\text{MG}|}\sum_{i=1}^{|\text{MG}|}\mathbf{X}_{L_C}\mathbf{W}_{\text{f}} + \mathbf{b}_{\text{f}}\big)$.

5 Pooling experiments

Tasks and setup We illustrate the applicability of the Mapper-GNN synthesis within a pooling framework by evaluating DMP and MPR in a variety of settings: social (IMDB-Binary, IMDB-Multi, Reddit-Binary, Reddit-Multi-5k), citation networks (Collab) and chemical data (D&D, Mutag, NCI1, Proteins) [Kersting et al., 2016]. We adopt a 10-fold cross-validation approach to evaluating the graph classification performance of DMP, MPR and other competitive state-of-the-art methods. The random seed was set to zero for all experiments (with respect to dataset splitting, shuffling and parameter initialisation), in order to ensure a fair comparison across architectures. All models were trained on a single Titan Xp GPU, using the Adam optimiser [Kingma and Ba, 2014] with early stopping on the validation set, for a maximum of 30 epochs. We report the classification accuracy using 95% confidence intervals calculated for a population size of 10 (the number of folds).

Baselines We compare the performance of DMP and MPR to two other pooling methods that we identify mathematical connections with: minCUT [Bianchi et al., 2019] and DiffPool [Ying et al., 2018]. Additionally, we include Graph U-Net [Gao and Ji, 2019] in our evaluation, as it has been shown to yield competitive results while performing pooling from the perspective of a learnable node ranking; we denote this approach by Top-k in the remainder of this section. The non-pooling baselines evaluated are the WL kernel [Shervashidze et al., 2011], a 'flat' model (2 MP steps and global average pooling) and an average-readout linear classifier. Full details of the model architectures and hyperparameters can be found in Appendix G.

Results and discussion The graph classification performance obtained by these models is reported in Table 1. We reveal that MPR ranks either first or second on all social datasets, or achieves accuracy scores within 0.5% of the best-performing model. This result confirms that PageRank-based pooling exploits the power-law distributions in this domain. The performance of DMP is similar on social data and generally higher on molecular graphs. We attribute this to the fact that all nodes in molecular graphs tend to have a similar PageRank score—MPR is therefore likely to assign all nodes to one cluster, effectively performing a readout. In this domain, DMP performs particularly well on Mutag, where it is second-best and improves by 3.7% over MPR, showing the benefits of having a differentiable lens in challenging data settings. Overall, MPR achieves the best accuracy on 2 datasets (D&D, Collab) and the next best result on 3 more (Proteins, Reddit-Binary and Reddit-Multi-5k). DMP improves on MPR by less than 1% on NCI1, Proteins, IDMB-Binary and IMDB-Multi, showing the perhaps surprising strength of the simple, fixed-lens pooling operator.

[2]Note that one or more {embedding $\rightarrow$ pooling} operations may be sequentially performed in the pipeline.

References

Peter W Battaglia, Jessica B Hamrick, Victor Bapst, Alvaro Sanchez-Gonzalez, Vinicius Zambaldi, Mateusz Malinowski, Andrea Tacchetti, David Raposo, Adam Santoro, Ryan Faulkner, et al. Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*, 2018.

Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Mincut Pooling in Graph Neural Networks. *arXiv preprint arXiv:1907.00481*, 2019.

Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):18–42, 2017.

Cătălina Cangea, Petar Veličković, Nikola Jovanović, Thomas Kipf, and Pietro Liò. Towards Sparse Hierarchical Graph Classifiers. *arXiv preprint arXiv:1811.01287*, 2018.

Mathieu Carrière and Steve Oudot. Structure and stability of the one-dimensional mapper. *Foundations of Computational Mathematics*, 18:1333–1396, 2018.

Mathieu Carriere, Bertrand Michel, and Steve Oudot. Statistical Analysis and Parameter Selection for Mapper. *The Journal of Machine Learning Research*, 19(1):478–516, 2018.

Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *ArXiv*, abs/2005.03675, 2020.

Frédéric Chazal and Bertrand Michel. An introduction to Topological Data Analysis: fundamental and practical aspects for data scientists. *arXiv preprint arXiv:1710.04019*, 2017.

Jim Demmel. UC Berkeley CS267 - Lecture 20: Partitioning Graphs without Coordinate Information II, 1995. URL https://people.eecs.berkeley.edu/~demmel/cs267-1995/lecture20/lecture20.html.

Tamal K. Dey, Facundo Mémoli, and Yusu Wang. Topological analysis of nerves, reeb spaces, mappers, and multiscale mappers. In *Symposium on Computational Geometry*, 2017.

Miroslav Fiedler. Algebraic Connectivity of Graphs. *Czechoslovak mathematical journal*, 23(2): 298–305, 1973.

Emden R Gansner and Stephen C North. An open graph visualization system and its applications to software engineering. *Software: practice and experience*, 30(11):1203–1233, 2000.

Hongyang Gao and Shuiwang Ji. Graph U-Nets. In *International Conference on Machine Learning*, pages 2083–2092, 2019.

Mustafa Hajij, Paul Rosen, and Bei Wang. Mapper on Graphs for Network Visualization, 2018.

Jingjia Huang, Zhangheng Li, Nannan Li, Shan Liu, and Ge Li. AttPool: Towards Hierarchical Feature Representation in Graph Convolutional Networks via Attention Mechanism. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 6480–6489, 2019.

Kristian Kersting, Nils M. Kriege, Christopher Morris, Petra Mutzel, and Marion Neumann. Benchmark Data Sets for Graph Kernels, 2016. URL http://graphkernels.cs.tu-dortmund.de.

Diederik P Kingma and Jimmy Ba. Adam: A Method for Stochastic Optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-Supervised Classification with Graph Convolutional Networks. *arXiv preprint arXiv:1609.02907*, 2016.

Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-Attention Graph Pooling. In *International Conference on Machine Learning*, pages 3734–3743, 2019.

Jure Leskovec. CS224W: Social and Information Network Analysis - Graph Clustering, 2016. URL http://snap.stanford.edu/class/cs224w-2016/slides/clustering.pdf.

Enxhell Luzhnica, Ben Day, and Pietro Liò. On graph classification networks, datasets and baselines. *ArXiv*, abs/1905.04682, 2019a.

Enxhell Luzhnica, Ben Day, and Pietro Lio. Clique pooling for graph classification. *arXiv preprint arXiv:1904.00374*, 2019b.

Yao Ma, Suhang Wang, Charu C Aggarwal, and Jiliang Tang. Graph Convolutional Networks with EigenPooling. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 723–731, 2019.

L. McInnes, J. Healy, and J. Melville. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. *ArXiv e-prints*, February 2018.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The PageRank Citation Ranking: Bringing Order to the Web. Technical report, Stanford InfoLab, 1999.

Ekagra Ranjan, Soumya Sanyal, and Partha Pratim Talukdar. ASAP: Adaptive Structure Aware Pooling for Learning Hierarchical Graph Representations. *arXiv preprint arXiv:1911.07979*, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The Graph Neural Network Model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective Classification in Network Data. *AI magazine*, 29(3):93–93, 2008.

Nino Shervashidze, Pascal Schweitzer, Erik Jan Van Leeuwen, Kurt Mehlhorn, and Karsten M Borgwardt. Weisfeiler-lehman graph kernels. *Journal of Machine Learning Research*, 12(77): 2539–2561, 2011.

Gurjeet Singh, Facundo Memoli, and Gunnar Carlsson. Topological Methods for the Analysis of High Dimensional Data Sets and 3D Object Recognition, 2007.

Joshua B. Tenenbaum, Vin de Silva, and John C. Langford. A global geometric framework for nonlinear dimensionality reduction. *Science*, 290(5500):2319–2323, 2000. ISSN 0036-8075. doi: 10.1126/science.290.5500.2319. URL https://science.sciencemag.org/content/290/5500/2319.

Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9:2579–2605, 2008. URL http://www.jmlr.org/papers/v9/vandermaaten08a.html.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep Graph Infomax. *arXiv preprint arXiv:1809.10341*, 2018.

Zhilin Yang, William W. Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. In *ICML*, 2016.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical Graph Representation Learning with Differentiable Pooling. In *Advances in Neural Information Processing Systems*, pages 4800–4810, 2018.

A Deep Graph Mapper

Here, we describe how to use DGM for visualisation, using the three main degrees of freedom of the Mapper algorithm to guide our discussion: the lens function, the cover, and the clustering algorithm.

The lens is a function $f : V \to \mathbb{R}^d$ over the vertices, which acts as a filter that emphasises certain features of the graph. Typically, d is a small integer—in our case, $d \in \{1, 2\}$. The choice of f depends on the graph properties that should be highlighted by the visualisation. In this work, we leverage the recent progress in the field of graph representation learning and propose a parameterised lens function based on graph neural networks (GNNs). We thus consider a function $f_\theta(v) = g_\theta(V, E, \mathbf{X})_v$, where g is a GNN with parameters θ taking as input a graph $G = (V, E)$ with n nodes and node features $\mathbf{X} \in \mathbb{R}^{n \times k}$. In practice, we often consider a function composition $f_\theta(v) = (r \circ g_\theta)_v$, where $r : \mathbb{R}^{n \times d'} \to \mathbb{R}^{n \times d}$ is a dimensionality reduction algorithm like t-SNE [van der Maaten and Hinton, 2008].

Unlike the traditional graph theoretic lens functions proposed by Hajij et al. [2018], GNNs can naturally learn to integrate the features associated with the graph and its topology, while also scaling computationally to large, complex graphs. Additionally, visualisations can flexibly be tuned for the task of interest by adjusting the lens g_θ through the associated loss function of the model.

The cover $\mathcal{U}$ determines the resolution of the visualisation. Here, unless otherwise mentioned, we use the usual cover choice for Mapper. When $d = 1$, we use a set of equally-sized overlapping intervals over the real line. When $d = 2$, this is generalised to a grid of overlapping cells in the real plane. Using more cells will produce more detailed visualisations, while higher overlaps between the cells will increase the connectivity of the output graph. When chosen suitably, these hyperparameters are a powerful mechanism for obtaining multi-scale visualisations.

The clustering algorithm statistically computes the (topological) connected components of the cover sets U_i. Mapper does not require a particular type of clustering algorithm. However, when the input topological space X is a graph, a natural choice, also adopted by Hajij et al. [2018], is to take the (graph) connected components of the subgraphs induced by the vertices $f^{-1}(U_i), i \in I$. This is also the approach we follow in this work.

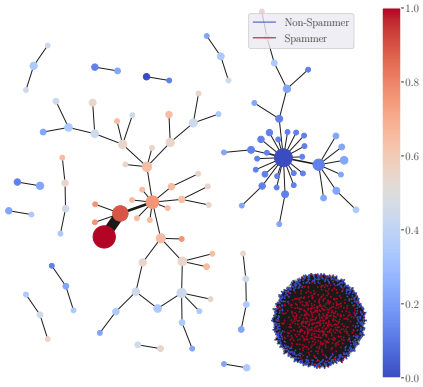


Figure 2: DGM visualisation of a synthetic spammer graph with 1k nodes and 50k edges, shown in full in the bottom right corner using Graphviz [Gansner and North, 2000]. The plot uses as the lens function a GNN predicting the probability of each node to be a spammer—indicated by the color-bar. DGM eliminates the visual clutter of the full graph and illustrates how spammers tend to form a large cluster at the core of the graph, while non-spammers form many smaller communities.

We refer to this instance of Mapper, with the choices described above, as Deep Graph Mapper (DGM). The algorithm is further explained step-by-step on a cartoon example in Figure 1. Pseudocode is also included in Appendix B.

We note once again that the edges of this graph reflect a semantic similarity between clusters. However, when visualising graphs, we often wish to have a **structure-oriented visualisation** that allows us to observe how different clusters that were formed by the refined pull back cover are connected. This problem can be exacerbated when using GNNs, since they may completely ignore the structure, if features are discriminative enough on their own [Luzhnica et al., 2019a]. To address this, in Appendix D we introduce and evaluate a variant called Structural Deep Graph Mapper (SDGM). Due to the limited space, we only include DGM visualisations in the main text.

A.1 Visualisations in Supervised Learning

The first application of DGM and SDGM is in a supervised learning context, where f_θ is trained via a cross entropy loss function to classify the nodes of the graph. When the classification is binary, $f_\theta : V \to [0, 1]$ outputs the probability that a node belongs to the positive class. This probability acts directly as the parameterisation of the graph nodes. An example is shown in Figure 2. For a multi-label classification problem, we first perform a dimensionality reduction on the logits of the classifier,

which reduces them to points in $\mathbb{R}^2$. We then visualise each class individually, by colouring the nodes based on the average predicted probability of the nodes in each cluster to belong to that class. An example is given for the Neural Networks class of Cora [Sen et al., 2008] in the left plot of Figure 3.

When labels are available, we also produce visualisations augmented with ground-truth information. These visualisations can provide a label-driven understanding of the graph. For instance, in Figure 4 we colour each node of the DGM visualisation for the Cora ML citation network according to the most frequent class in the corresponding cluster. This visualisation reveals many interesting properties of the graph. Genetic Algorithms seem to be the most isolated subdomain, but a line of work merging them with Reinforcement Learning and Neural Networks exists, as depicted in the top-left corner of the figure. At the same time, Neural Networks are interacting with many other ML subfields, especially Probabilistic Methods and Reinforcement Learning.

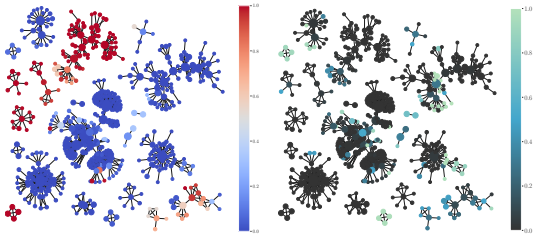


Figure 3: Visualisation coloured with the average predicted probability for the 'neural network' class in the Cora citation network (left) and the absolute difference between the predicted probability and the class occurrence frequency in each cluster (right). The right plot clearly identifies systematic mistakes of the GCN classifier (in lighter tones).

These visualisations can additionally be used to find connected clusters that contain systematically misclassified nodes and see how they relate to others. To that end, we colour the nodes based on the absolute difference between the average predicted probability for a class and the normalised occurrence of that class within the corresponding clusters. This can be seen on the right side of Figure 3, where the lighter tones indicate a higher discrepancy between these two scalars.

All the examples in this section use a Graph Convolutional Network (GCN) [Kipf and Welling, 2016] with four layers trained to classify the nodes of the graph. For the spammer graph, the lens is given by the predicted spam probability of each node and the cover consists of 10 intervals over $[0, 1]$, with 30% overlap. For the Cora graph, we use a 2D t-SNE projection of the network logits and a cover formed of 64 equally-sized cells, with 20% overlap across each dimension. We refer the reader to Appendix E for a more extensive set of visualisations.

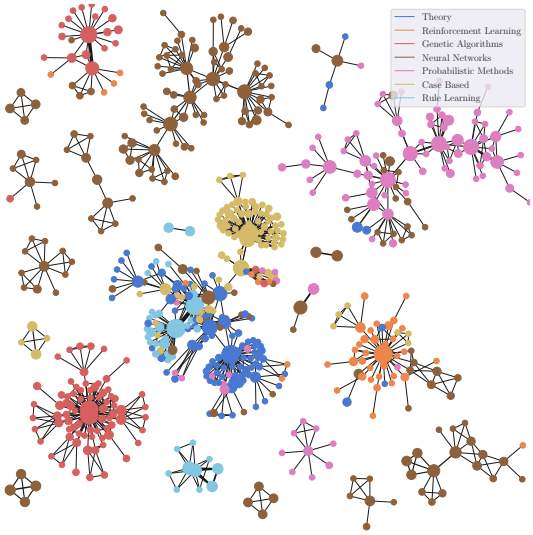
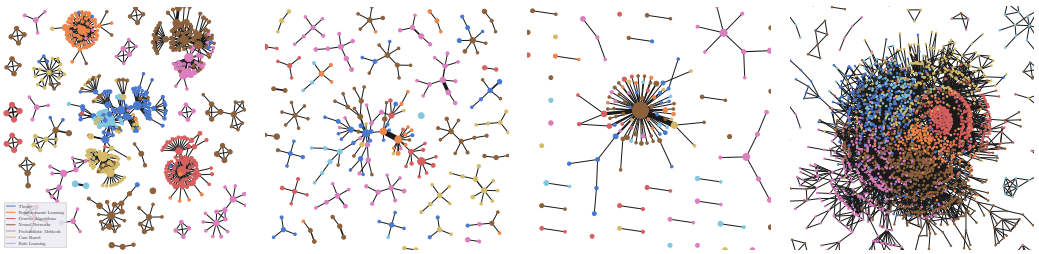


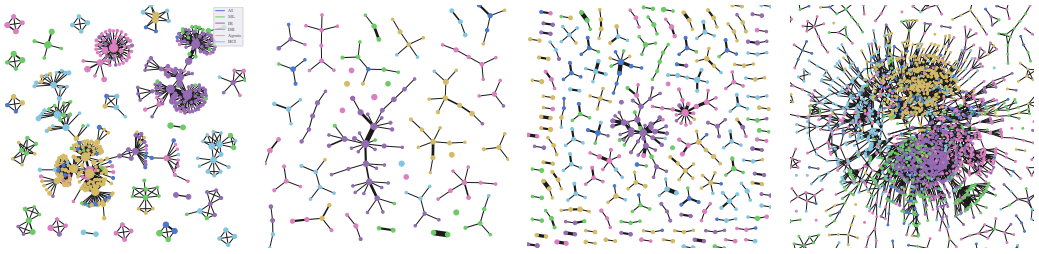
Figure 4: Visualisation of the Cora ML citation network, where the nodes are coloured based on the most frequent class in the corresponding cluster.

A.2 Visualisation in Unsupervised Learning

The second application corresponds to an unsupervised learning scenario, where the challenge is obtaining a parameterisation of the graph in the absence of labels. This is the typical use case for unsupervised graph representation learning models [Chami et al., 2020]. The approach we follow is to train a model to learn node embeddings in $\mathbb{R}^{d'}$ (in our experiments, $d' = 512$), which can be reduced, as before, to a low-dimensional space via a dimensionality reduction method r. Unsupervised visualisations can be found in the qualitative evaluation in Section A.3; more can be found in Appendix E.



(a) DGM on Cora (b) Density on Cora (c) PageRank on Cora (d) Cora full



(e) DGM on CiteSeer (f) Density on CiteSeer (g) PageRank on CiteSeer (h) CiteSeer full

Figure 5: Qualitative comparison between DGM (a, e), Mapper with an RBF graph density function [Hajij et al., 2018] (b, f), and Mapper with a PageRank function [Hajij et al., 2018] (c, g). The Graphviz visualisation of the graph cores (d, h) are added for reference. The first and second row show plots for Cora and CiteSeer, respectively. The graphs are coloured based on the most frequent class in each cluster to aid the comparison. DGM with unsupervised lens implicitly makes all dataset classes appear in the visualisation clearly separated. This does not happen in the baseline visualisations, which mainly focus on the class with the highest number of nodes from each graph.

A.3 Qualitative Evaluation

In this section, we qualitatively compare DGM against the two best-performing graph theoretic lens functions proposed by Hajij et al. [2018], on the Cora and CiteSeer [Sen et al., 2008] citation networks. Namely, we compare against a PageRank [Page et al., 1999] lens function and a graph density function $f(v) = \sum_{u \in V} \exp(-D(u, v)/\delta)$, where D is the distance matrix of the graph. For DGM, we use a composition of an unsupervised Deep Graph Infomax (DGI) [Veličković et al., 2018] model $g_\theta : V \to \mathbb{R}^{512}$ and a dimensionality reduction function $r : \mathbb{R}^{512} \to \mathbb{R}^2$ based on t-SNE. To aid the comparison, we mark the nodes with the colour of the most frequent class in the corresponding cluster. Additionally, we include a Graphviz [Gansner and North, 2000] plot of the full graph. We carefully fine-tuned the covers for each combination of model and graph; these hyperparameters can be found in Appendix B. Appendix E extends this evaluation with additional graphs and an ablation study considering alternative unsupervised models, dimensionality reduction algorithms and alternative cover choices.

As depicted by Figure 5, DGM successfully summarises many of the properties of the graphs that are also reflected by full graph visualisations. For instance, on Cora, Genetic Algorithms (in red) are shown, as before, to be primarily isolated from other subdomains of the citation network. At the same time, related classes that largely overlap in the full visualisation—Probabilistic Methods and Neural Networks (NNs) on Cora or Information Retrieval (IR) and ML on CiteSeer—are connected in the DGM plot. In contrast, the baselines do not have the same level of granularity and fail to capture many such properties. Both PageRank and the graph density function tend to focus on the classes with the highest number of nodes, such as the IR class on CiteSeer or the NNs class on Cora, while largely de-emphasising other classes.

Naturally, the proposed visualisations also have certain limitations. In an unsupervised learning setting, in the absence of any labels or attributes for colouring the graph, the nodes have to be coloured based on a colormap associated with the abstract embedding space (shown in Appendix E), thus affecting the interpretability of the visualisations. In contrast, even though the graph theoretic lens functions produce lower quality visualisations, their semantics are clearly understood mathematically.

This is, however, a drawback shared even by some of the most widely used data visualisation methods, such as t-SNE or UMAP McInnes et al. [2018].

B DGM Model Details

In this section, we provide additional details about the DGM implementation and explain our cover choice for the qualitative evaluation in Section A.2.

B.1 DGM pseudocode

In Algorithm 1 we include the pseudocode for DGM. For simplicity, we assume that the lens is a neural network mapping vertices to numbers in $[0, 1]$, similarly to the GNN predicting the probability of a node to be spam from Section A.1. However, this can be easily extended to handle the other types of lenses and covers discussed in this work.

Algorithm 1: Deep Graph Mapper

Input : Graph $G = (V, E)$ with node features $\mathbf{X}$
Lens function $f_\theta : V \to [0, 1]$ with $f_\theta(v) = g_\theta(\mathbf{X}, v, E)_v$, where g_θ is a trained GNN
Cover $\mathcal{U}$ of $[0, 1]$ formed of n overlapping intervals U_i.
Output : A graph $M = (V', E')$

```
// compute the pull back cover
```
1 **foreach** *node* $v \in V$ **do**
2 **foreach** $i \in I$ **do**
3 **if** $f_\theta(v) \in U_i$ **then**
4 Add v to the preimage $f_\theta^{-1}(U_i)$

```
// compute the refined pull back cover
```
5 Initialise empty graph $M = (V', E')$
6 **foreach** $i \in I$ **do**
7 Let G_i be the subgraph of G induced by the nodes in $f_\theta^{-1}(U_i)$
8 **foreach** *connected component* c *of* G_i **do**
9 Create a node u and add it to V'
10 $s \leftarrow 0$
11 **foreach** *node* $v \in c$ **do**
 // save the set of nodes in G corresponding to node $u \in V'$
12 Add v to $u.nodes$
 // save the soft cluster that node v was assigned to
13 Add u to $v.clusters$
14 $s \leftarrow s + f_\theta(v)$
15 $u.colour \leftarrow \frac{s}{|u.nodes|}$

```
// compute the 1-skeleton of the nerve
```
16 **foreach** *node* $v \in V$ **do**
17 **foreach** *unordered pair of nodes* (a, b) *with* $a, b \in v.clusters$ **do**
18 **if** $a \neq b$ **then**
19 Add the undirected edge $e = (a, b)$ to E' if not added already
20 $e.weight \leftarrow e.weight + 1$

We note that the colour of the nodes can flexibly be set based on other aggregation schemes other than the mean lens value as in line 15. For instance, when we visualise a single class in the multi-label classification context, the colour is given by the average predicted probability of that class for all the nodes in $u.nodes$ for each node $u \in V'$.

We note that the complexity of the algorithm is linear in the size of the input graph, assuming the maximum number of overlapping intervals and the number of intervals are (small) constants. Alternatively, the final step in lines 16–20 can also be computed in a vectorised form, with the help

of a soft cluster assignment matrix $\mathbf{S} \in \mathbb{R}^{|V| \times |V'|}$. Then, the adjacency matrix of the output graph is given by $\mathbf{S}^T\mathbf{S}$.

B.2 Cover details for Section A.2

In this section, we detail our cover choices for the unsupervised experiments in Section A.2. For the PageRank lens, we used $n = 26$ intervals with overlap 0.15 on Cora and $n = 30$ intervals on CiteSeer with the same amount of overlap. For the RBF density lens, we used 20 intervals with 0.2 overlap both on Cora and CiteSeer. The scale of the RBF kernel was set to 3 and 4, respectively, for the two graphs. For the DGM visualisation, we used a grid of 64 2D cells on Cora and 36 cells on Citeseer, with an overlap of 0.2 in each direction of the 2D space on both graphs.

C Proof of Proposition 4.1

Throughout the proof we consider a graph $G(V, E)$ with self loops for each of its N nodes. The self-loop assumption is not necessary, but it elegantly handles a number of degenerate cases involving nodes isolated from the other nodes in its cluster. We refer to the edges of a node which are not a self-loop as **external edges**. Additionally, for simplicity we assume each of the soft clusters is connected, and therefore the pull back cover is the same as the refined pull back cover. The proof can easily be modified to take this extra case into account by either considering an instance of Mapper where the clustering algorithm assigns all nodes to the same cluster (same as no clustering), or the expanded graph G' (introduced below) can be constructed with additional edges which make each soft cluster connected.

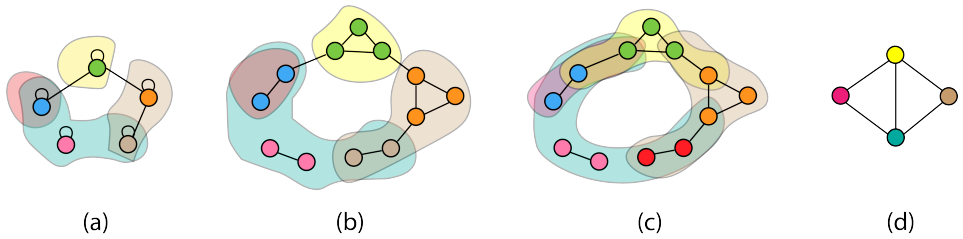


Figure 6: An illustration of the proof for Proposition 4.2. Connecting the clusters connected by an edge in the original graph (a) is equivalent to constructing an expanded graph with an expanded cluster assignment (b), doing a 1-hop expansion of the expanded cluster assignment (c) and finally taking the 1-skeleton of the nerve (d) of the cover obtained in (c).

Let $s : V \to \triangle_{K-1}$ be a soft cluster assignment function that maps the vertices to the $(K-1)$-dimensional unit simplex. We denote by $s_k(v)$ the probability that vertex v belongs to cluster $k \leq K$ and $\sum_k^K s_k(v) = 1$. This function can be completely specified by a cluster assignment matrix $\mathbf{S} \in \mathbb{R}^{N \times K}$ with $S_{ik} = s_k(i)$. This is the soft cluster assignment matrix computed by minCut and Diff pool via a GCN.

Definition C.1. Let $G(V, E)$ be a graph with self-loops for each node. The expanded graph $G'(V', E')$ of G is the graph constructed as follows. For each node $v \in V$ with at least one external edge, there is a clique of nodes $\{v_1, \ldots, v_{deg(v)}\} \subseteq V'$. For each external edge $(v, u) \in E$, a pair of nodes from their corresponding cliques without any edges outside their clique are connected. Additionally, isolated nodes become in the new graph two nodes connected by an edge.

Essentially, the connections between the nodes in the original graph are replaced by the connections between the newly formed cliques such that every new node is connected to at most one node outside its clique. An example of an expanded graph is shown in Figure 6 (b). The nodes in the newly formed cliques are coloured similarly to node from the original graph they originate from.

Definition C.2. Let $G(V, E)$ be a graph with self loops for each node, s a soft cluster assignment function for it, and $G'(V', E')$ the expanded graph of G. Then the expanded soft cluster assignment s^* is a cluster assignment function for G', where $s_k^*(v_i) = s_k(v)$ for all the nodes $v_i \in V'$ in the corresponding clique of $v \in V$.

In plain terms, all the nodes in the clique inherit through s^* the cluster assignments of the corresponding node from the original graph. This is also illustrated by the coloured contours of the expanded graph in Figure 6 (b).

Definition C.3. Let $\boldsymbol{S}$ be a soft cluster assignment matrix for a graph $G(V, E)$ with adjacency matrix $\boldsymbol{A}$. The 1**-hop expansion** of assignment $\mathbf{S}$ with respect to graph G is a new cluster assignment function $s' : V \to \triangle_{K-1}$ induced by the row-normalised version of the cluster assignment matrix $\mathbf{S}' = \mathbf{AS}$.

As we shall now prove, the 1-hop expansion simply extends each soft cluster from $\mathbf{S}$ by adding its 1-hop neighbourhood as in Figure 6 (c).

Lemma C.1. An element of the soft cluster assignment matrix $S'_{i,k} \neq 0$ if and only if node i is connected to a node j (possibly $i = j$), which is part of the soft cluster k of the assignment induced by $\mathbf{S}$.

Proof. By definition $S'_{ik} = \sum_j^N A_{ij} S_{jk} = 0$ if and only if $A_{ij} S_{jk} = 0$, for all j. This can happen if and only if $A_{ij} = 0$ (nodes i and j are not connected by an edge) or $S_{jk} = 0$ (node j does not belong to soft cluster k defined by $\mathbf{S}$) for all j. Therefore, $S'_{ik} \neq 0$ if and only if there exists a node j such that i is connected to j and j belongs to soft cluster k defined by $\mathbf{S}$. □

Corollary C.1.1. Nodes that are part of a cluster k defined by $\mathbf{S}$, are also part of k under the assignment $\mathbf{S}'$.

Proof. This immediately follows from Lemma C.1 and the fact that each node has a self-loop. □

Lemma C.2. The adjacency matrix $\mathbf{A}' = \mathbf{S^T AS}$ defines a new graph where the clusters induced by S are connected if and only if there is a common edge between them.

Proof. Let $\mathbf{L} = \mathbf{AS}$. Then, $A'_{ij} = \sum_k^N S^T_{ik} L_{kj} = 0$ if and only if $S^T_{ik} = 0$ (node k does not belong to cluster i) or $L_{kj} = 0$ (node k is not connected to any node belonging to cluster j by Lemma C.1), for all k. Therefore, $A'_{ij} \neq 0$ if and only if there exists a node k such that k belongs to cluster i and k is connected to a node from cluster j. □

This result shows that soft cluster assignment methods connect clusters that have at least one edge between them. We will use this result to show that a Mapper construction obtains an identical graph.

Let s' be the 1-hop expansion of the expanded soft cluster assignment of graph G'. Let the soft clusters induced by s' be $\mathcal{A} = \{A_1, A_2, \ldots, A_K\}$. Additionally, let $\mathcal{B} = \{B_1, B_2, \ldots, B_K\}$ be an open cover of $\triangle_{K-1}$ with $B_k = \{\mathbf{x} \in \triangle_{K-1} | \mathbf{x}_k > 0\}$. Then the pull back cover induced by $(s', \mathcal{B})$ is $\mathcal{A}$ since $s'^{(-1)}(B_k) = A_k$ (i.e. all nodes with a non-zero probability of belonging to cluster k).

Lemma C.3. Two clusters A_x and A_y have a non-empty intersection in the expanded graph if and only if their corresponding clusters C_x, C_y in the original graph have a common edge between them.

Proof. If direction: By Corollary C.1.1, the case of a self edge becomes trivial. Now, let $v \in C_x$ and $u \in C_y$ be two nodes connected by an external edge in the original graph. Then in the expanded graph $G'(V', E')$, there will be clique nodes $v_i, u_i \in V'$ such that $(v_i, u_i) \in E'$. By taking the 1-hop expansion of the extended cluster assignment, both v_i and v_j will belong to A_x, A_y by Lemma C.1, since they are in each other's 1-hop neighbourhood. Since we have chosen the clusters and the nodes arbitrarily, this proves this direction.

Only if direction: Let C^*_x and C^*_y be the (expanded) clusters in the expanded graph corresponding to the clusters C_x and C_y in the original graph. Let node v_i be part of the non-empty intersection between soft clusters A_x and A_y defined by $\boldsymbol{S}'$ in the expanded graph G'. By Lemma C.1, v_i belongs to A_x if and only if there exists $v_j \in V'$ such that $(v_i, v_j) \in E'$ and $v_j \in C^*_x$. Similarly, there must exist a node $v_k \in V'$ such that $(v_i, v_k) \in E'$ and $v_k \in C^*_y$. By the construction of G', either both v_j, v_k are part of the clique v_i is part of, or one of them is in the clique, and the other is outside the clique.

Suppose without loss of generality that v_j is in the clique and v_k is outside the clique. Then, $v_i \in C^*_x$ since they share the same cluster assignment. By the construction of G' the edge between the

corresponding nodes of v_i and v_k in the original graph G is an edge between C_x and C_y. Similarly, if both v_j and v_k are part of the same clique with v_i, then they all originate from the same node $v \in C_x, C_y$ in the original graph. The self-edge of v is an edge between C_x to C_y. □

Proposition C.1. Let $G(V, E)$ be a graph with self loops and adjacency matrix $\mathbf{A}$. The graphs defined by $sk_1(\mathcal{N}(\mathcal{A}))$ and $\mathbf{S}^T\mathbf{AS}$ are isomorphic.

Proof. Based on Lemma C.3, $sk_1(\mathcal{N}(\mathcal{A}))$ connects two soft clusters in G defined by $\boldsymbol{S}$ if and only if there is a common edge between them. By Lemma C.2, soft cluster assignment methods connect the soft clusters identically through the adjacency matrix $\mathbf{A}' = \mathbf{S^T AS}$. Therefore, the unweighted graphs determined by $sk_1(\mathcal{N}(\mathcal{A}))$ and $\mathbf{A}'$ are isomorphic. □

Note that our statement refers strictly to unweighted edges. However, an arbitrary weight function $w : V \to \mathbb{R}$ can easily be attached to the graph obtained though $sk_1(\mathcal{N}(\mathcal{A}))$.

D Structural Deep Graph Mapper

The edges between the nodes in a DGM visualisation denote semantic similarities discovered by the lens. However, it is sometimes desirable that the connectivity of the graph is explicitly accounted for in the visualisations, being involved in more than simply computing the refined pull back cover. Including structural aspects explicitly is particularly important when using GNNs as the lens function. As shown by Luzhnica et al. [2019a], if the features are discriminative enough, GNNs can completely exclude any structural information from the semantics of the embedding space.

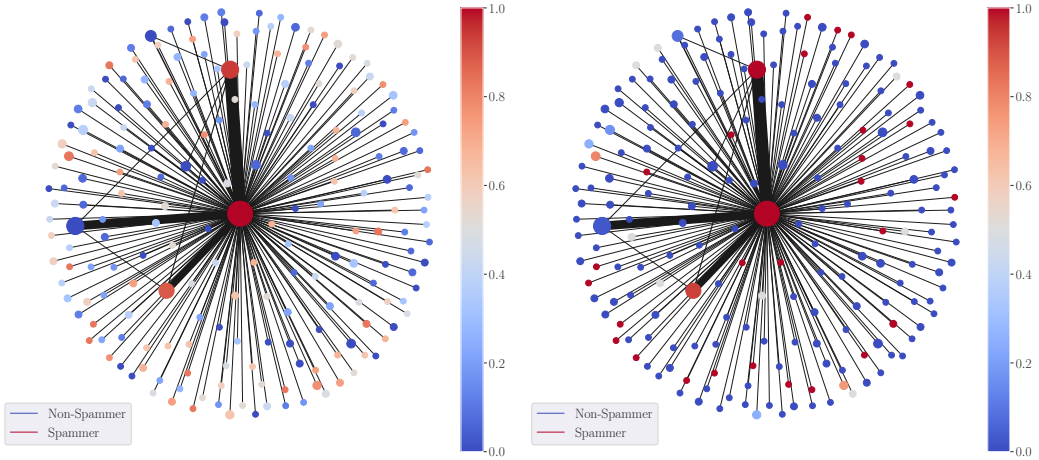
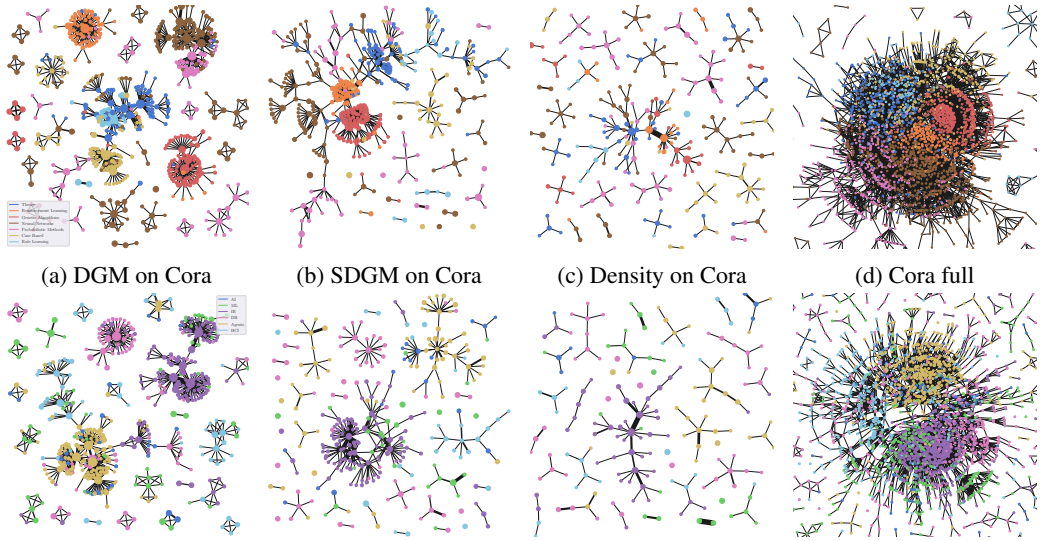


Figure 7: SDGM visualisation with colour indicating spam probability (left) and the labeled visualisation indicating the proportion of spammers in each cluster (right). The thickness of the edges is now proportional to the number of edges between clusters. We used a filtration value of $\epsilon = 0.01$ and $g = 0$ for the overlap. This visualisation also illustrates that the spammers are densely connected to the other nodes in the graph, while non-spammers form smaller connected groups. However, unlike the DGM visualisation, this graph also shows the (structural) edges between the spammers and the non-spammers.

Therefore, with this motivation in mind, we also propose **Structural DGM (SDGM)**, a version of DGM that connects the elements of the refined pull back cover based on the number of edges between the clusters, as opposed to the number of nodes as in DGM. To do so, we rely on the proof from Appendix C, and essentially apply DGM in the expanded graph G'. SDGM uses the refined pull back cover induced by the 1-hop expansion in the expanded graph (see Appendix C) to compute the nerve. However, a downside of this approach is that the output graph may often be too dense to visualise. Therefore, we use a weighting function $w : E \to [0, 1]$ to weight the edges of the resulting graph and a filtration value $\epsilon \in [0, 1]$. We then filter out all the edges e with $w(e) < \epsilon$, where $w(e)$ is determined by the normalised weighted adjacency matrix $\mathbf{S^T AS}$ denoting the (soft) number of edges between two clusters. For SDGM, the cover overlap parameter g effectively sets a trade-off between

the number of structural and semantic connections in the SDGM graph. For $g = 0$, only structural connections exist. At the same time, the filtration constant ϵ is an additional parameter that adjusts the resolution of the visualisations. Higher values of ϵ result in sparser graphs, while lower values increase the connectivity between the nodes. The pseudocode for SDGM is given in Algorithm 2.



(a) DGM on Cora (b) SDGM on Cora (c) Density on Cora (d) Cora full

(e) DGM on CiteSeer (f) SDGM on CiteSeer (g) Density on CiteSeer (h) CiteSeer full

Figure 8: Qualitative comparison of SDGM (second column) and the other Mapper lens functions from Section A.2.

In Figure 7, we include an SDGM visualisation for the spammer graph using the same neural network described in Section A.1 for the DGM visualisation of the same graph from Figure 2. Unlike the DGM visualisation, the SDGM ones contains connections between spammers and non-spammer communities, indicating that even though these nodes are semantically opposed, they are evidently connected in the graph. At the same time, the visualisation indicates more clearly how the spammer nodes form the core of the graph and how the non-spammers for smaller communities at the edge of the graph.

Additionally, we extend the qualitative comparison from Section A.2 with the SDGM visualisations in Figure 8. For the SDGM visualisation we use the same embeddings as for DGM, and set the number of cells to 400, overlap to zero, and ϵ (the edge filtration value) to 0.05. Even though SDGM reveals the same binary relations as DGM, the former is able to capture more interesting structures in the graph, such as the loops formed at the core of the graph on Cora or the loop formed by the papers on Information Retrieval and ML on CiteSeer.

E Additional Visualisations

In this section, we present additional visualisations and ablation studies.

E.1 Ablation study for dimensionality reduction

We study how the choice of the dimensionality reduction method for the unsupervised visualisations affects the visualisation. To test this, we consider the following dimensionality reduction methods: t-SNE [van der Maaten and Hinton, 2008], UMAP [McInnes et al., 2018], IsoMap [Tenenbaum et al., 2000] and PCA. We use the same model as in Section A.2 and 81 2D cells for the cover of all models. The overlap was set after fine-tuning to 0.2 for t-SNE and UMAP, and to 0.1 for the other two models. Figure 9 displays the four visualisations. As expected, t-SNE and UMAP produce more visually-pleasing outputs, due to their superior ability to capture variation in the GNN embedding space. However, the features highlighted by all visualisations are largely similar, indicating the same binary relations between clusters. This demonstrates that the GNN embedding space is robust to the choice of the dimensionality reduction method.

Algorithm 2: Structural Deep Graph Mapper

Input : Graph $G = (V, E)$ with node features $\mathbf{X}$
Lens function $f_\theta : V \to [0, 1]$ with $f_\theta(v) = g_\theta(\mathbf{X}, v, E)_v$, where g_θ is a trained GNN
Cover $\mathcal{U}$ of $[0, 1]$ formed of n overlapping intervals U_i

Output : A graph $M = (V', E')$

`// compute the pull back cover`

foreach *node* $v \in V$ **do**
  foreach $i \in I$ **do**
    if $f_\theta(v) \in U_i$ **then**
      Add v to the preimage $f_\theta^{-1}(U_i)$

`// compute the refined pull back cover`

Initialise empty graph $M = (V', E')$
foreach $i \in I$ **do**
  Let G_i be the subgraph of G induced by the nodes in $f_\theta^{-1}(U_i)$
  foreach *connected component* c *of* G_i **do**
    Create a node u and add it to V'
    `// save the set of nodes in` G `corresponding to node` $u \in V'$
    $s \leftarrow 0$
    foreach *node* $v \in c$ **do**
      Add v to $u.nodes$
      Add u to $v.clusters$
      $s \leftarrow s + f_\theta(v)$
    $u.color \leftarrow \frac{s}{|u.nodes|}$

`// compute the 1-skeleton of the nerve`

foreach *edge* $(u, v) \in E$ **do**
  foreach *unordered mapper node pair* (u', v') *with* $u' \in u.clusters, v' \in v.clusters$ **do**
    if $u' \neq v'$ **then**
      Add the undirected edge $e = (u', v')$ to E' if not added already
      $e.weight \leftarrow e.weight + 1$

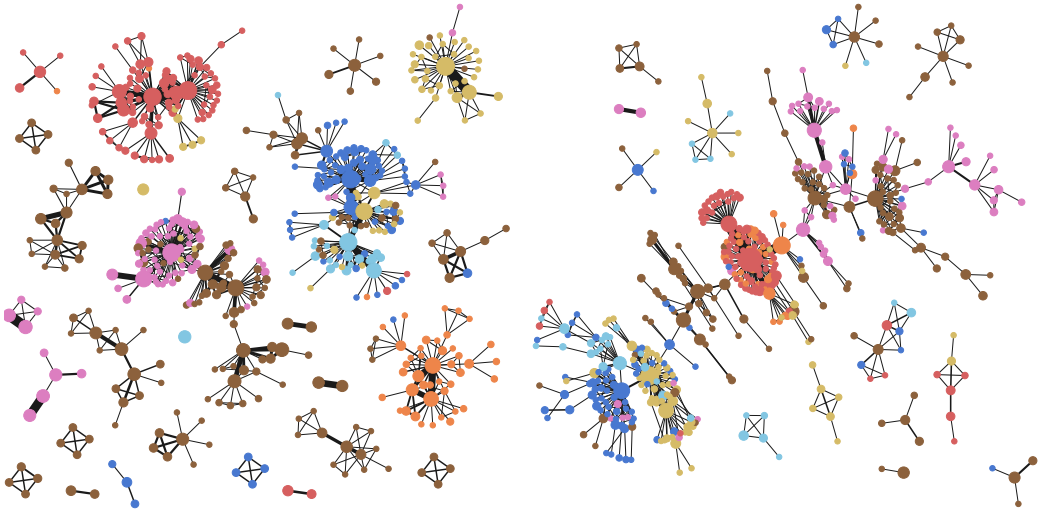
E.2 Ablation for the unsupervised lens

To better understand the impact of GNNs on improving the quality of the Mapper visualisations, we perform an ablation study on the type of unsupervised lens functions used within Mapper. The first model we consider is simply the identity function taking as input only graph features. The second model is a randomly initialised DGI model, as in Section A.2. Despite the apparent simplicity of a randomly initialised model, it was shown that such a method produces reasonably good embeddings, often outperforming other more sophisticated baselines [Veličković et al., 2018]. Ultimately, we use our trained DGI model from Section A.2. For all these models, we perform a t-SNE reduction of their embedding space to obtain a 2D output space and use 81 overlapping cells that cover this space. An overlap of 0.2 is used across all models.

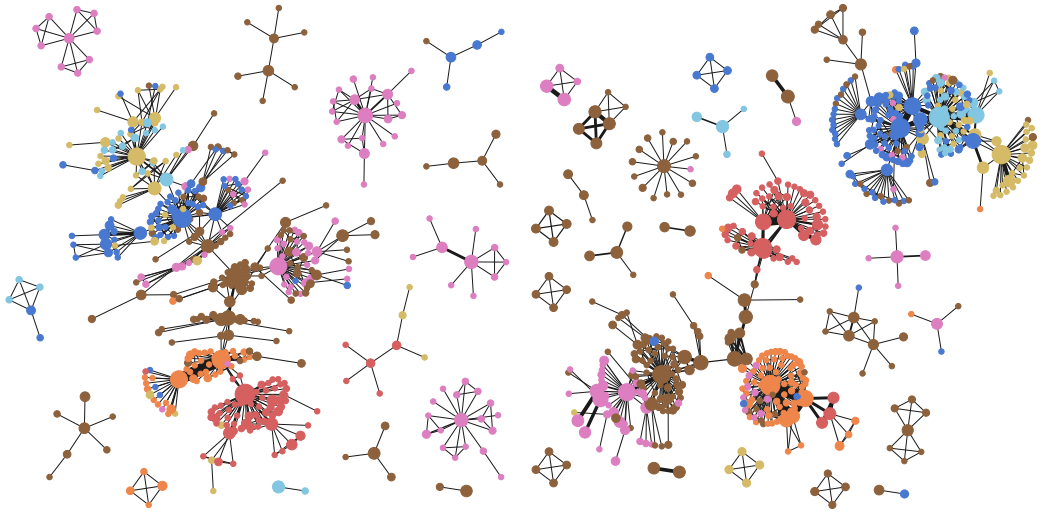
The three resulting visualisations are depicted in Figure 10. The identity model and the untrained DGI model are only able to find a limited number of relations between clusters, mainly between the light blue and dark blue classes. At the same time, the untrained DGI model clusters some of the other classes slightly better and forms bigger connected components. However, none of these models do particularly well. In contrast, the trained DGI model makes all the classes prominently show up in the visualisation, together with their main interactions.

E.3 The PubMed graph

Our qualitative comparison involving unsupervised lens relied on the Cora and CiteSeer citation networks. In this section we consider a much larger citation network, PubMed [Yang et al., 2016], containing over 19717 nodes and 44338 edges. In contrast, Cora has 2708 nodes and 5278 edges,



(a) t-SNE (b) PCA

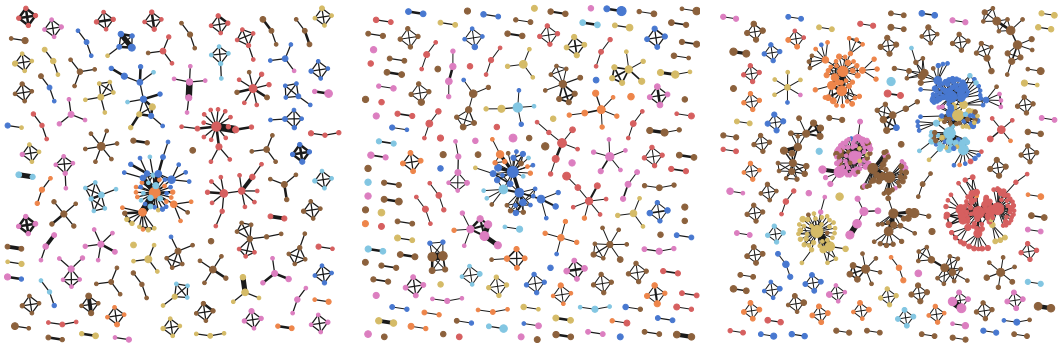


(c) Isomap (d) UMAP

Figure 9: Ablation for dimensionality reduction methods. While t-SNE and UMAP produce slightly better visualisations, the graph features displayed by the visualisations are roughly consistent across all of the four dimensionality reduction techniques.

while CiteSeer has 3327 nodes and 4552 edges. Due to the large size of this dataset, we were forced to use an approximation for the distance matrix of the graph, only relying on four levels of depth to compute the RBF density lens from Section A.2. Similarly, this time we supply for reference only a sample subgraph, taken from the largest connected component of the graph, since the entire graph is too large to visualise.

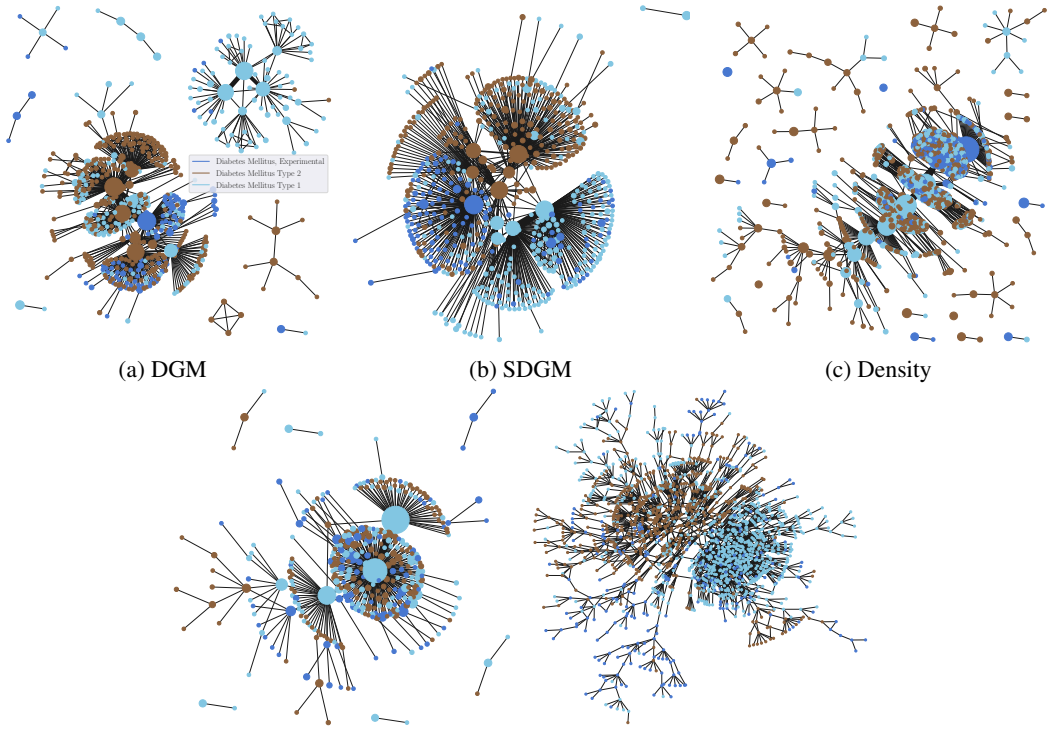
For DGM and SDGM, we use the model from Section A.2 with 64 cells as the cover, and overlap 0.1 and 0.05, respectively. In both cases, we perform a dimensionality reduction using UMAP to project the 512-dimensional DGI embeddings to a 2D space. For the RBF density lens, we use 20 intervals with overlap 0.2, while for the PageRank lens we use the same number of intervals, but 0.15 overlap. As shown in Figure 11, the graph theoretic lens proposed by Hajij et al. [2018] struggle in this setting, since the three classes of the dataset (denoting publications about different types of diabetes) become very mixed. In contrast, the DGM and SDGM visualisations capture the main properties that can also be seen from the full depiction of the main connected component of the graph. Namely, the light blue class (Diabetes Type 1) tends to be more isolated from the other two classes and it is mostly connected with the brown-coloured class denoting Diabetes Type 2. At the same



(a) Identity (b) Untrained DGI model (c) Trained DGI model

Figure 10: Ablation for different types of unsupervised lenses. The trained DGI model significantly improves the quality of the visualisations.

time, the visualisation shows that the dark blue class (Diabetes, Experimental) is mainly connected with the brown-coloured class.



(a) DGM (b) SDGM (c) Density

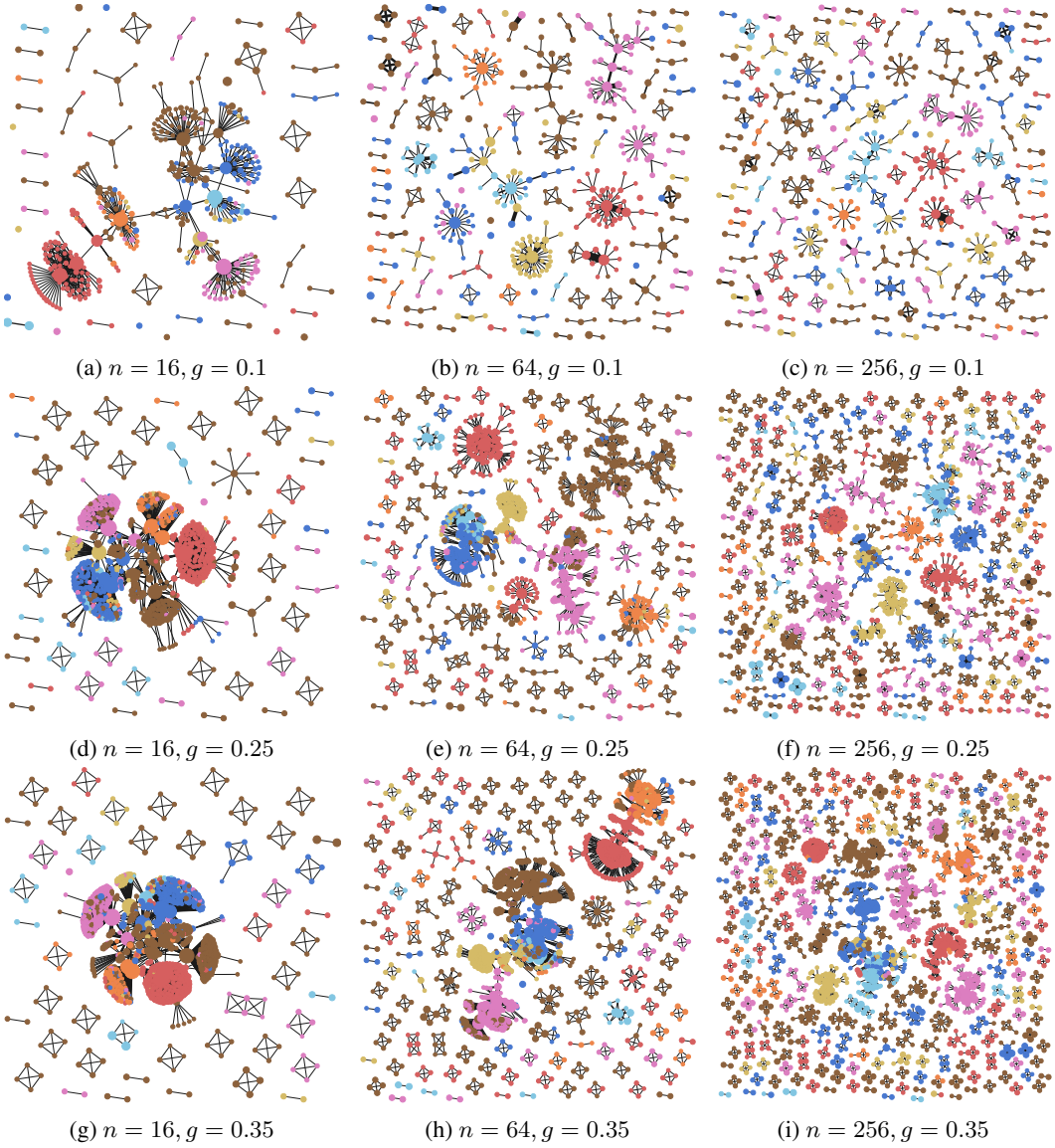
(d) PageRank (e) Graphviz Core

Figure 11: Qualitative comparison of different Mapper lens functions on the PubMed citation network.

E.4 Hierarchical visualisations

One of the most powerful features of Mapper is the ability to produce multi-resolution visualisations through the flexibility offered by the cover hyperparameters. As described in Section A, having a higher number of cells covering the output space results in more granular visualisations containing more nodes, while a higher overlap between these cells results in increased connectivity. We highlight these trade-offs in Figure 12, where we visualise the Cora citation network using 9 combinations of cells and overlaps. These kinds of hierarchical visualisations can help one identify what are the persistent features of the graph. For instance, when inspecting the plots that use $n = 64$ cells, it

is visible that the connections between the light blue class and the yellow class persist for all 3 degrees of overlap, indicating that this is a persistent feature of the graph. In contrast, the connection between the red and orange classes is relatively reduced ($g = 0.25$) or none ($g = 0.1$) for low values of overlap, but it clearly appears at $g = 0.35$ in the top-right corner, suggesting that the semantic similarity between the two classes is very scale-sensitive (that is, less persistent).



(a) $n = 16, g = 0.1$ (b) $n = 64, g = 0.1$ (c) $n = 256, g = 0.1$

(d) $n = 16, g = 0.25$ (e) $n = 64, g = 0.25$ (f) $n = 256, g = 0.25$

(g) $n = 16, g = 0.35$ (h) $n = 64, g = 0.35$ (i) $n = 256, g = 0.35$

Figure 12: Hierarchical visualisations of the Cora citation network using various number of cover cells and degrees of overlap.

E.5 Abstract colormaps

As mentioned in Section A.2, in the absence of any labels or attributes to colour the graph with, we fill in the nodes of the graph using an abstract colour space associated with the embedding space. The resulting colourisation shows which area of the embedding space the cluster comes from. An unsupervised visualisation of Cora (as in Section A.2, but this time using the abstract 2D color-space) is shown in Figure 13.

E.6 High-resolution supervised visualisations

In addition to the higher-resolution version of the Cora supervised visualisation from Section A.1, we provide two other supervised visualisations of CiteSeer and PubMed. These can be found in Figures 15, 16 and 17. For all models, we use the same architecture as in Section A.1.

F Mapper and Spectral Clustering

The relationship between Mapper for graphs and spectral clustering has been observed by Hajij et al. [2018]. This link is a strong indicator that Mapper can compute 'useful' clusters for pooling. We formally restate this observation below and provide a short proof.

Proposition F.1. Let L be the Laplacian of a graph $G(V, E)$ and l_2 the eigenvector corresponding to the second lowest eigenvalue of L, also known as the Fiedler vector [Fiedler, 1973]. Then, for a function $f : V \to \mathbb{R}, f(v) = l_2(v)$, outputting the entry in the eigenvector l_2 corresponding to node v and a cover $\mathcal{U} = \{(-\infty, \epsilon), (-\epsilon, +\infty)\}$, Mapper produces a spectral bi-partition of the graph for a sufficiently small positive ϵ.

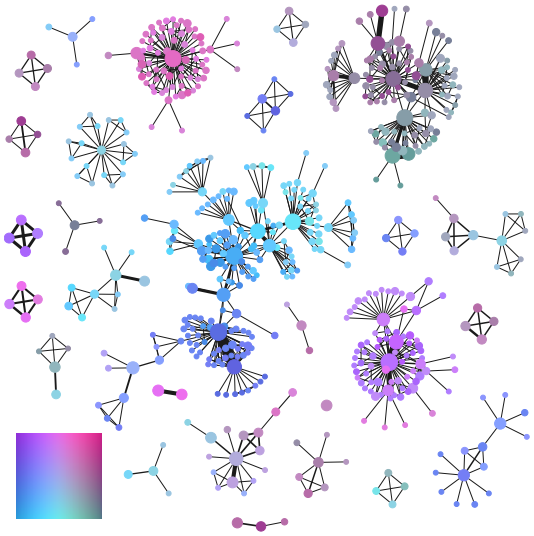


Figure 13: Cora visualisation colored using the abstract color-space associated with the embedding space (show in the bottom left corner).

Proof. It is well known that the Fiedler vector can be used to obtain a 'good' bi-partition of the graph based on the signature of the entries of the vector (i.e. $l_2(v) > 0$ and $l_2(v) < 0$) (please refer to Demmel [1995] for a proof). Therefore, by setting ϵ to a sufficiently small positive number $\epsilon < \min_v |l_2(v)|$, the obtained pull back cover is a spectral bi-partition of the graph. □

The result above indicates that Mapper is a generalisation of spectral clustering. As the latter is strongly related to min-cuts [Leskovec, 2016], the proposition also links them to Mapper.

G Model Architecture and Hyperparameters

We optimise both DMP and MPR with respect to the cover cardinality n, the cover overlap (δ for DMP, overlap percentage g for MPR), learning rate and hidden size. The Top-k architecture is evaluated using the code provided in the official repository, where separate configurations are defined for each of the benchmarks. The minCUT architecture is represented by the sequence of operations described by Bianchi et al. [2019]: *MP(32)-pooling-MP(32)-pooling-MP(32)-GlobalAvgPool*, followed by a linear softmax classifier. The *MP(32)* block represents a message-passing operation performed by a graph convolutional layer with 32 hidden units $\mathbf{X}^{(t+1)} = \text{ReLU}(\tilde{\mathbf{A}}\mathbf{X}^{(t)}\mathbf{W}_m + \mathbf{X}^{(t)}\mathbf{W}_s)$, where $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}\mathbf{A}\mathbf{D}^{-\frac{1}{2}}$ is the symmetrically-normalised adjacency matrix and $\mathbf{W}_m, \mathbf{W}_s$ are learnable weight matrices representing the message passing and skip-connection operations within the layer. The DiffPool model follows the same sequence of steps.

The optimised models have the following configurations:

- DGM—learning rate $5e^{-4}$, hidden sizes $\{128, 128\}$ and:
  - D&D and Collab: cover sizes $\{20, 5\}$, interval overlap 10%, batch size 32;
  - Proteins: cover sizes $\{8, 2\}$, interval overlap 25%, batch size 128;
  - Reddit-Binary: cover sizes $\{20, 5\}$, interval overlap 25%, batch size 32;

- Top-k—specific dataset configurations, as provided in the official GitHub repository[3] (`run_GUNet.sh`);
- minCUT—learning rate $1e^{-3}$, same architecture as reported by the authors in the original work [Bianchi et al., 2019];
- DiffPool—learning rate $1e^{-3}$, hidden size 32, two pooling steps, pooling ratio $r = 0.1$, global average mean readout layer, with the exception of Collab and Reddit-Binary, where the hidden size was 128.

We additionally performed a hyperparameter search for DiffPool on hidden sizes $32, 64, 128$ and for DGM, over the following sets of possible values:

- all datasets: cover sizes $\{[40, 10], [20, 5]\}$, interval overlap $\{10\%, 25\%\}$;
- D&D: learning rate $\{5e^{-4}, 1e^{-3}\}$;
- Proteins: learning rate $\{2e^{-4}, 5e^{-4}, 1e^{-3}\}$, cover sizes $\{[24, 6], [16, 4], [12, 3], [8, 2]\}$, hidden sizes $\{64, 128\}$.

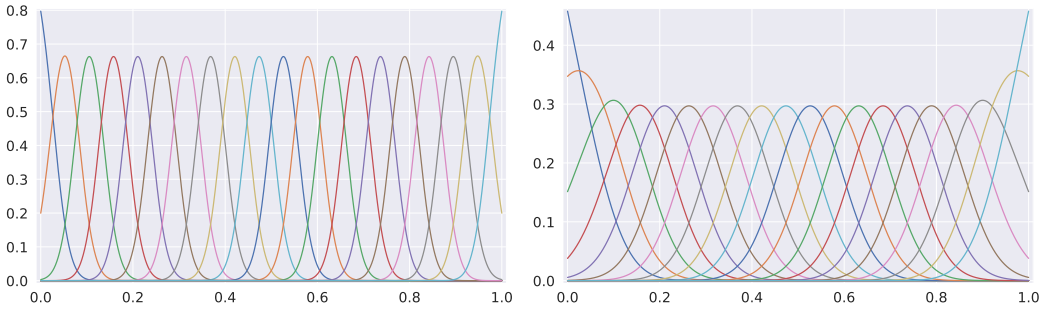


Figure 14: Two covers of RBF kernels with different scales $\delta = 0.001$ and $\delta = 0.01$.

G.1 The RBF kernel cover

In Figure 14 we show two examples of RBF kernel covers for the output space. The scale of the kernel, δ, determines the amount of overlap between the cover elements. At bigger scales, there is a higher overlap between the clusters as depicted in the two plots. Because the line is one-dimensional, a point on the unit interval can only be part of a small number of clusters (i.e. the kernels for which the value is greater than zero), assuming the scale δ is not too large. Therefore, DMP can be seen as a DiffPool variant where the low-entropy constraint on the cluster assignment is satisfied topologically, rather than by a loss function enforcing it.

[3] github.com/HongyangGao/Graph-U-Nets

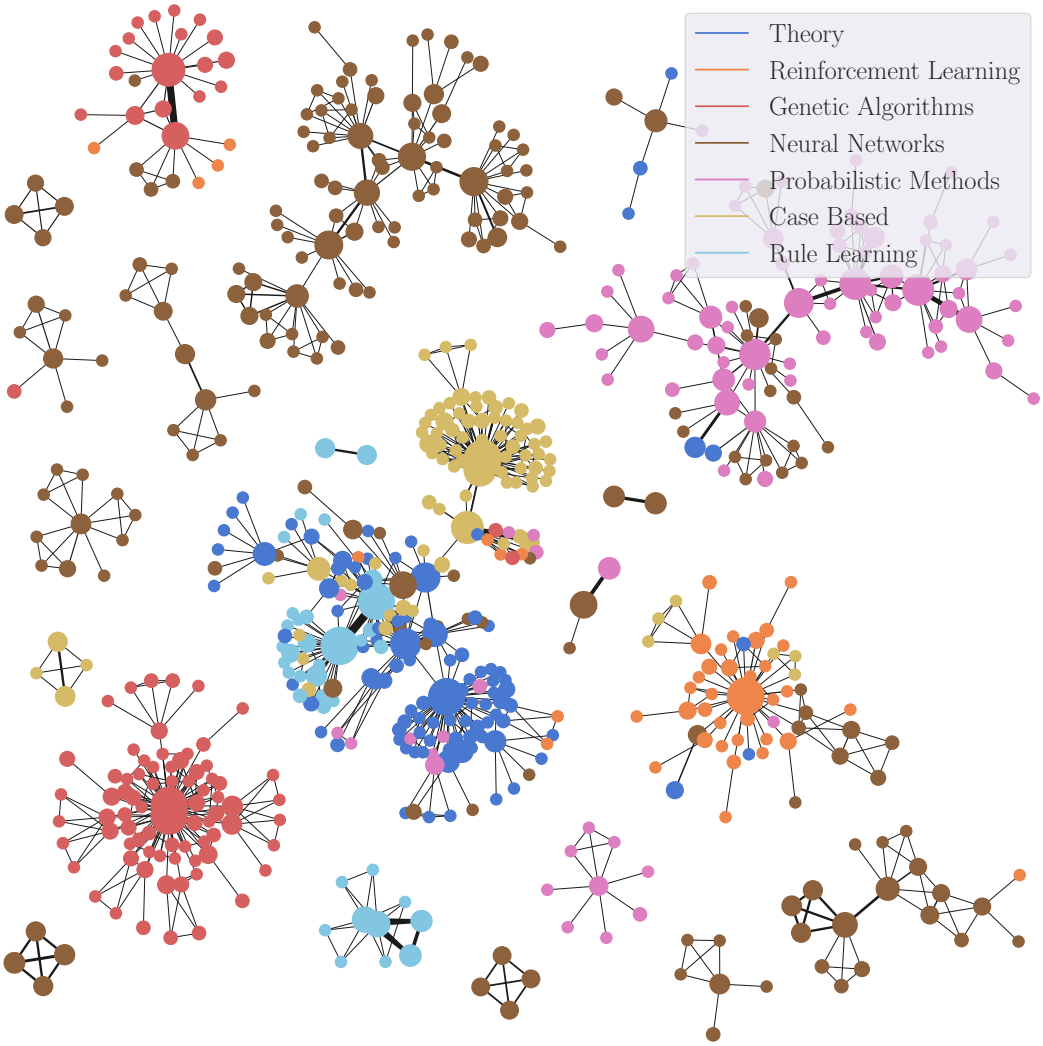


Figure 15: Cora using a GCN classifier as the lens.

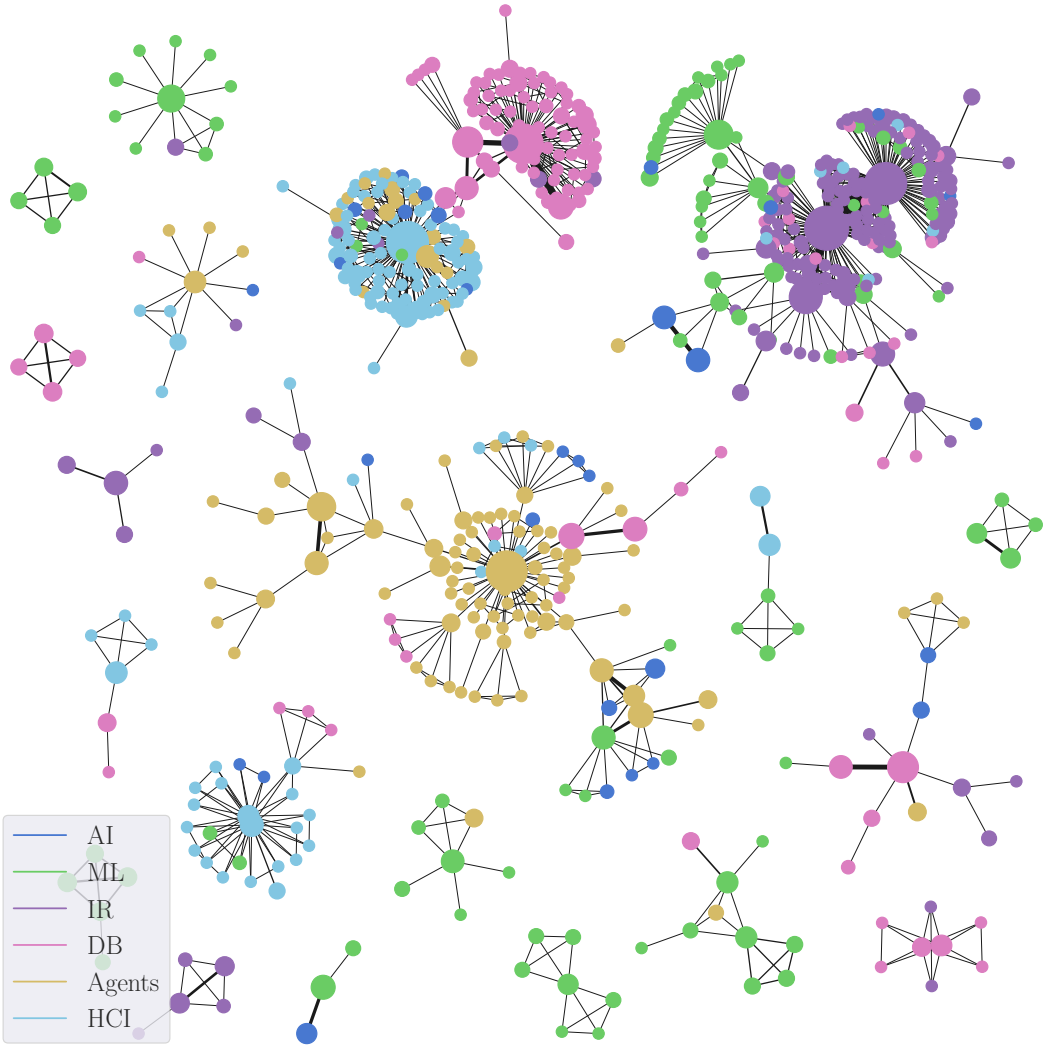


Figure 16: Citeseer using a GCN classifier as the lens.

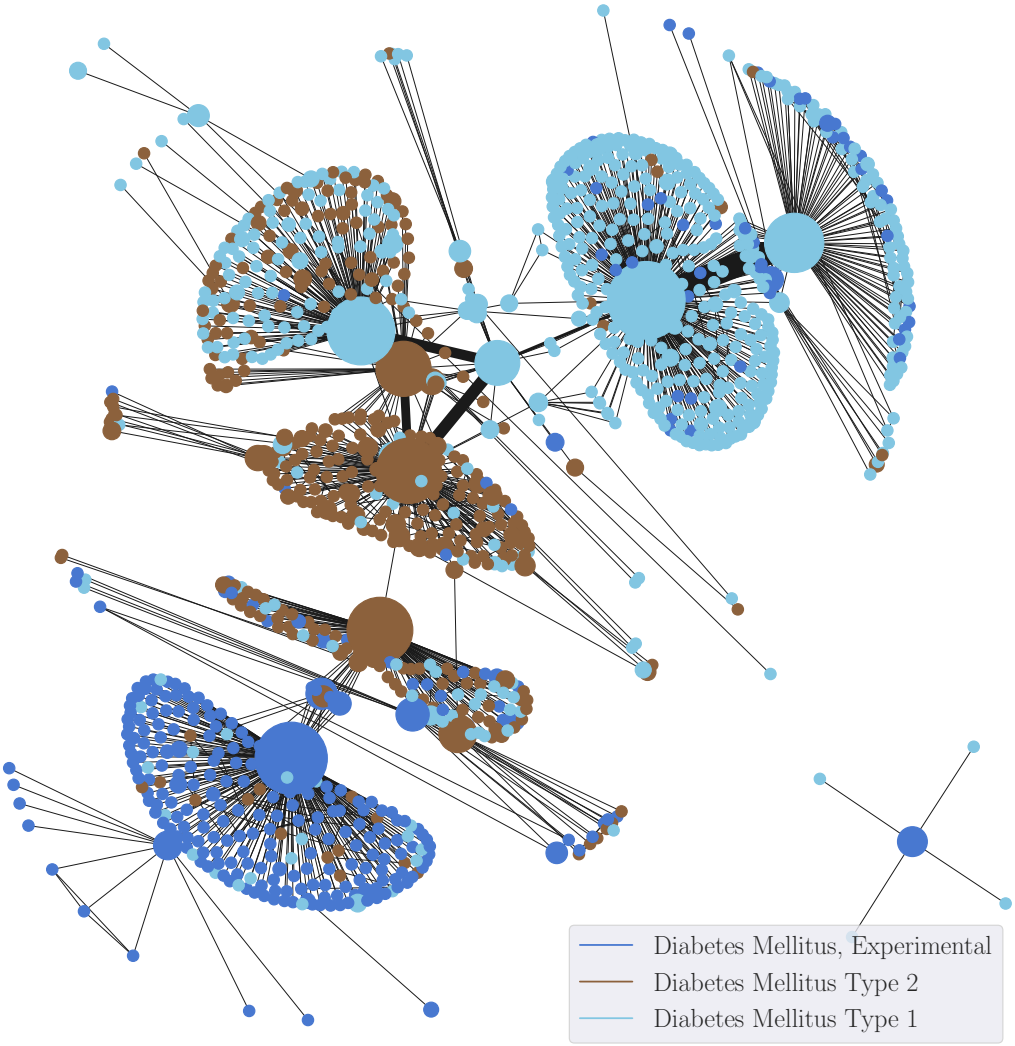


Figure 17: PubMed using a GCN classifier as the lens.