The Map Equation Goes Neural: Mapping Network Flows with Graph Neural Networks

Christopher Blöcker[*]
Data Analytics Group
Department of Informatics
University of Zurich, Switzerland
christopher.bloecker@uzh.ch

Chester Tan
Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science
Julius-Maximilians-Universität Würzburg, Germany
chester.tan@uni-wuerzburg.de

Ingo Scholtes[†]
Chair of Machine Learning for Complex Networks
Center for Artificial Intelligence and Data Science
Julius-Maximilians-Universität Würzburg, Germany
ingo.scholtes@uni-wuerzburg.de

Abstract

Community detection is an essential tool for unsupervised data exploration and revealing the organisational structure of networked systems. With a long history in network science, community detection typically relies on objective functions, optimised with custom-tailored search algorithms, but often without leveraging recent advances in deep learning. Recently, first works have started incorporating such objectives into loss functions for deep graph clustering and pooling. We consider the map equation, a popular information-theoretic objective function for unsupervised community detection, and express it in differentiable tensor form for optimisation through gradient descent. Our formulation turns the map equation compatible with any neural network architecture, enables end-to-end learning, incorporates node features, and chooses the optimal number of clusters automatically, all without requiring explicit regularisation. Applied to unsupervised graph clustering tasks, we achieve competitive performance against state-of-the-art deep graph clustering baselines in synthetic and real-world datasets.

1 Introduction

Many real-world networked systems are organised in communities: groups of nodes that are more similar to each other than to the rest. Communities provide insights into network structure at the mesoscale, revealing sub-systems by analysing link patterns. Motivated by different research questions, several characterisations of what constitutes "good" communities have been proposed [1, 2], however, neither of them is fundamentally more correct than any other. Moreover, no single community-detection method outperforms all others on any given network [3], motivating the ongoing efforts of research on community detection. Typically, community-detection approaches formulate an objective function that calculates a quality score for a given partition of the network's nodes into communities. Finding the best partition is an NP-hard search problem and often involves custom heuristic algorithms that attempt to minimise their objective function [4, 5, 6].

[*] Also at Chair of Machine Learning for Complex Networks, Center for Artificial Intelligence and Data Science, Julius-Maximilians-Universität Würzburg, Germany

[†] Also at Data Analytics Group, Department of Informatics, University of Zurich, Switzerland

38th Conference on Neural Information Processing Systems (NeurIPS 2024).

https://doi.org/10.52202/079017-0554

Graph neural networks (GNNs) have enabled applying deep learning to graph-structured data by utilising the input graph as the neural network's computational graph [7, 8, 9]. Typical tasks for GNNs include node labelling, graph labelling, and link prediction, all of which involve learning meaningful representations jointly from the graph's topology, the nodes' features, and, possibly, the edges' features. Graph labelling relies on coarse-graining the graph through identifying groups of "similar" nodes and aggregating their links and features, also referred to as pooling [10, 11, 12], which is related to graph clustering [13], however, these two tasks have different goals.

While GNNs excel at incorporating node and edge features with graph topology, including this information is also possible but more challenging with traditional network science approaches, typically requiring modelling or adjusting objective functions and their optimisation algorithms. On the other hand, objective functions for community detection provide precise interpretations as to why one partition is considered better than another while deep-learning-based approaches are black boxes. Model selection in deep learning is often done through regularisation techniques or cross-validation; in contrast, objective functions that are based on the minimum description length (MDL) principle naturally implement Occam's razor, preventing overfitting and enabling principled model selection without requiring extra regularisation or cross-validation [14, 15].

Here, we combine the benefits of traditional community-detection approaches and deep learning and consider the map equation, an information-theoretic objective function for community detection [16]. By adapting the map equation for soft cluster assignments and implementing it in differentiable tensor form, we enable end-to-end optimisation of the map equation as a loss function with gradient descent and GNNs. In analogy to the map equation's stochastic optimisation algorithm Infomap [6], we call our approach Neuromap and evaluate it against Infomap and several recent GNN-based graph clustering methods. Applied to synthetic and real-world networks, Neuromap demonstrates competitive performance against recent deep graph clustering baselines.

Our key contributions can be summarised as follows:

1. We adapt the map equation as a differentiable loss function for end-to-end deep graph clustering and propose Neuromap, a deep-learning-based alternative to the popular Infomap algorithm for unsupervised community detection with the map equation. Neuromap is compatible with any neural network architecture, detects overlapping communities, leverages node features for improved performance on real-world networks, and, by following the minimum description length principle, does not require explicit regularisation.
2. We extensively evaluate Neuromap on hundreds of synthetic and ten real datasets against recent baselines paired with various neural network architectures. Neuromap outperforms the baseline on the synthetic networks in most settings and is amongst the best performers in seven out of ten real datasets.
3. By choosing a higher maximum number of clusters than previous works, we show empirically that recent baselines tend to overfit and report considerably more than the ground-truth number of communities. Moreover, we find that choosing a small maximum number of communities is often detrimental to graph clustering performance.

2 Related work

Community detection. *Communities*, also called *clusters* or *modules*, are groups of nodes that are more "similar" to each other than to the rest, often understood as having more links inside than between groups [1, 2]. However, this rather general characterisation leaves precise details of what constitutes a community open. Modularity compares the observed link densities inside communities against a randomised version of the network [17]. The stochastic block model and its variants assume a latent block structure where the probability that two nodes are connected depends only on their block memberships [18, 19]. The map equation identifies communities as regions where a random walker tends to stay for a relatively long time [16, 20]. Traditional clustering approaches, such as k-means, group nodes based on their proximity in space, however, here we consider identifying communities from the link patterns in networked systems. For a detailed overview of community detection in complex networks, we refer to [1, 2].

Minimum description length principle. The minimum description length principle (MDL) is a model-selection approach that formalises Occam's razor and frames learning as a compression

https://doi.org/10.52202/079017-0554

problem [14, 15]. MDL states that the best model for data D is the one that compresses the data the most. In traditional MDL, the data's two-part description length $L(D) = \min_M L(M) + L(D \mid M)$ is the smallest achievable length over all models M, where $L(M)$ is the model's description length, and $L(D \mid M)$ is the data's description length, given the model. MDL has been adopted for a wide range of applications, including regularising neural networks' weights [21], investigating deep neural networks' data-compression capabilities [22], analysing the characteristics of datasets [23], and community detection [16, 24].

Deep graph clustering and pooling. Graph clustering has long been a research focus in machine learning [13, 25]. Spectral approaches cluster, for example, the eigenspace of a graph's Laplacian matrix or identify communities through graph cuts [26, 27]. Methods based on neural embeddings involve learning node representations with, for example, DeepWalk [28] or node2vec [29], followed by applying standard clustering approaches such as k-means, assuming that similar nodes are embedded at similar locations [30, 31]. Other approaches include graph autoencoders [32, 33], contrastive learning [34], and self-expressiveness [35]. Recently, minimum cuts [11, 12], modularity [36, 37], and the Bernoulli-Poisson model [38] have been integrated with GNNs as loss functions for graph pooling and clustering. Such GNN-based approaches can incorporate graph structure as well as node and edge features in end-to-end optimisation of the clustering objective. Inspired by pooling in convolutional neural networks, graph pooling coarse-grains links, node features, and edge features to summarise graphs, enabling GNNs with improved performance on node and graph classification tasks [10, 12]. Consequently, graph pooling has become a research focus for GNNs, emphasising the importance of graph clustering as a primary objective [10, 12, 36]. For recent surveys of deep graph clustering, we refer to [39, 40].

3 Background: the map equation

The map equation [16, 20] is an information-theoretic objective function for unsupervised community detection that follows the MDL principle [14], and has demonstrated high performance in synthetic and real networks from across domains [41, 42, 43]. The map equation formulates community detection as a compression problem and uses random walks as a proxy to model dynamic processes on networks, also called *flow*. The goal is to describe the random walk as efficiently as possible by minimising its expected per-step description length – also called *codelength* – by partitioning the network into groups of nodes, called *modules*, where the random walker tends to stay for a relatively long time. In practice, however, the map equation does not simulate random walks; instead, the codelength is calculated analytically.

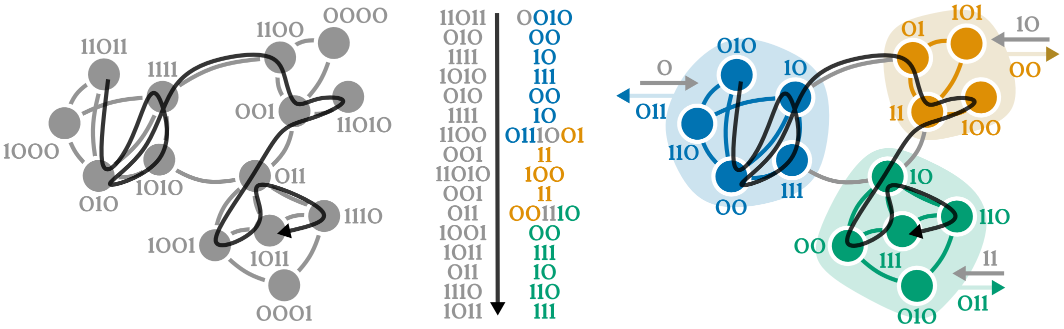


Figure 1: Coding principles behind the map equation. Colours indicate modules, codewords are shown next to nodes, and the black trace shows a sequence of random-walker steps. **Left:** All nodes belong to the same module and all codewords are unique. Encoding the random walk sequence requires 60 bits, or 3.72 bits per step in the limit. **Right:** Partitioning the network enables reusing codewords across modules, reducing the codelength. However, for a unique encoding, we need to introduce codewords for entering and exiting modules, shown next to the arrows pointing into and out of the modules. With this modular coding scheme, we can compress the description to 48 bits, or 3.01 bits per step in the limit. **Middle:** The two encodings of the random walker's steps.

Let $G = (V, E)$ be a graph with nodes V, links E, and let $w_{uv} \in \mathbb{R}_0^+$ denote the non-negative link weight on the link from node u to v. When all nodes are assigned to the same module, the codelength is defined by the Shannon entropy H over the nodes' visit rates [44], $H(P) = -\sum_{u \in V} p_u \log_2 p_u$, where p_u is node u's visit rate and $P = \{p_u \mid u \in V\}$ is the set of node visit rates. In undirected graphs, we compute visit rates directly as $p_u = s_u / \sum_{v \in V} s_v$, where $s_u = \sum_{v \in V} w_{uv}$ is node u's strength. In directed graphs, we compute the visit rates numerically with smart teleportation [45] and a power iteration. When we partition the nodes into modules, the codelength becomes a weighted average of the modules' entropies and the entropy at the so-called index level for switching between modules. Figure 1 illustrates the coding principle behind the map equation using Huffman codes [46]; note, however, that these codewords are only for illustration and we only care about their expected length in the limit to evaluate the map equation.

Minimising the map equation means balancing between small modules to achieve low module-level entropies and large modules for low index-level entropy. This trade-off between module- and index-level entropies prevents trivial solutions where all nodes are assigned to the same module or each node is assigned to a singleton module [14]. The map equation calculates the codelength for a partition M,

$$L(\mathsf{M}) = qH(Q) + \sum_{\mathsf{m} \in \mathsf{M}} p_{\mathsf{m}} H(P_{\mathsf{m}}). \tag{1}$$

Here, $q = \sum_{\mathsf{m}} q_{\mathsf{m}}$ is the random walker's module entry rate, $q_{\mathsf{m}} = \sum_{u \notin \mathsf{m}} \sum_{v \in \mathsf{m}} p_u t_{uv}$ is module m's entry rate, and $Q = \{q_{\mathsf{m}}/q \mid \mathsf{m} \in \mathsf{M}\}$ is the set of normalised module entry rates; $p_{\mathsf{m}} = \mathsf{m}_{\text{exit}} + \sum_{u \in \mathsf{m}} p_u$ is the rate at which the random walker moves in module m, including the module exit rate $\mathsf{m}_{\text{exit}} = \sum_{u \in \mathsf{m}} \sum_{v \notin \mathsf{m}} p_u t_{uv}$, and $P_{\mathsf{m}} = \{\mathsf{m}_{\text{exit}}/p_{\mathsf{m}}\} \cup \{p_u/p_{\mathsf{m}} \mid u \in \mathsf{m}\}$ is the set of normalised node visit and exit rates for module m. The random walker's transition probability from node u to v is $t_{uv} = w_{uv} / \sum_{v \in V} w_{uv}$. We can rewrite the map equation as (see Appendix A)

$$L(\mathsf{M}) = q \log_2 q - \sum_{\mathsf{m} \in \mathsf{M}} q_{\mathsf{m}} \log_2 q_{\mathsf{m}} - \sum_{\mathsf{m} \in \mathsf{M}} \mathsf{m}_{\text{exit}} \log_2 \mathsf{m}_{\text{exit}} - \sum_{u \in V} p_u \log_2 p_u + \sum_{\mathsf{m} \in \mathsf{M}} p_{\mathsf{m}} \log_2 p_{\mathsf{m}}. \tag{2}$$

The map equation framework has been extended for overlapping communities through state-space expansions with higher-order network models [47, 48], avoiding over-partitioning in sparse networks using a Bayesian regularisation approach [49], and to deal with sparse constrained structures [50]. Moreover, the map equation framework can incorporate node features through an extension [51] or by preprocessing data [52]. Detecting communities relies on Infomap [6], a greedy stochastic search algorithm that optimises the map equation. However, each of the above extensions requires preprocessing the input data, adjusting the loss function, or adapting the search algorithm. In contrast, adapting the map equation as a loss function for optimisation with gradient descent does not require any custom algorithm, thus enabling flexible experimentation with variations, scalability to GPU clusters, and incorporating it into other loss functions.

4 The map equation goes neural

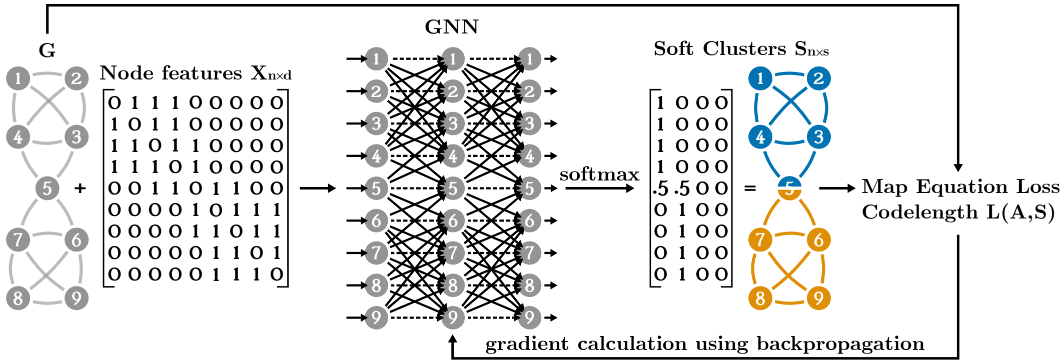
We set out to detect communities by optimising the map equation with GNNs through gradient descent, which essentially means learning coarse-graining node representations in the form of communities (Figure 2). While the standard map equation considers hard clusters where each node is assigned to exactly one module, we introduce a soft cluster assignment matrix $\mathbf{S}_{n \times s}$ to make the map equation differentiable and enable overlapping clusters. We optimise $\mathbf{S} = \operatorname{softmax}(\text{GNN}_\theta(\mathbf{A}, \mathbf{X}))$ indirectly by optimising the GNN's parameters θ, that is, its weights, with respect to the codelength L. Here, $\mathbf{A}_{n \times n}$ is the graph's adjacency matrix, $\mathbf{X}_{n \times d}$ is the node features matrix, $n = |V|$ is the number of nodes, s is the maximum allowed number of clusters, and d is the node feature dimension.

Without loss of generality, we assume directed networks. We denote the graph's total weight as $w_{\text{tot}} = \sum_{i \in V} \sum_{j \in V} w_{ij}$. Let $\mathbf{T}_{n \times n}$ be the random walker's transition matrix and $\mathbf{d}^{\text{in}}$ be the vector of weighted node in-degrees with

$$\mathbf{T}_{ij} = \begin{cases} \dfrac{w_{ij}}{\sum_{j \in V} w_{ij}} & \text{if } \sum_{j \in V} w_{ij} > 0, \\ 0 & \text{otherwise,} \end{cases} \qquad \mathbf{d}^{\text{in}}_j = \sum_{i \in V} w_{ij}.$$

To compute the vector $\mathbf{p}$ of node visit rates, we use smart teleportation [45] and the power iteration method: With probability α, the random walker teleports to a random node, chosen proportionally to

https://doi.org/10.52202/079017-0554


G
Node features $X_{n\times d}$
+
GNN
softmax
Soft Clusters $S_{n\times s}$
=
Map Equation Loss
Codelength L(A,S)
gradient calculation using backpropagation
1
2
3
4
5
6
7
8
9

$$\begin{bmatrix} 0&1&1&1&0&0&0&0&0\\ 1&0&1&1&0&0&0&0&0\\ 1&1&0&1&1&0&0&0&0\\ 1&1&1&0&1&0&0&0&0\\ 0&0&1&1&0&1&1&0&0\\ 0&0&0&0&1&0&1&1&1\\ 0&0&0&0&1&1&0&1&1\\ 0&0&0&0&0&1&1&0&1\\ 0&0&0&0&0&1&1&1&0 \end{bmatrix} \qquad \begin{bmatrix} 1&0&0&0\\ 1&0&0&0\\ 1&0&0&0\\ 1&0&0&0\\ .5&.5&0&0\\ 0&1&0&0\\ 0&1&0&0\\ 0&1&0&0\\ 0&1&0&0 \end{bmatrix}$$

Figure 2: Illustration of the setup for GNN-based community detection with the map equation. We learn soft cluster assignments $\mathbf{S}$ from the graphs adjacency matrix $\mathbf{A}$ and the node features $\mathbf{X}$. Here, we allow up to four clusters. When no node features are available, we set $\mathbf{X} = \mathbf{A}$.

the nodes' in-degrees, or follows a link with probability $1 - \alpha$. This approach leads to the iterative update rule $\mathbf{p}^{(t+1)} \leftarrow \frac{\alpha}{w_{\text{tot}}}\mathbf{d}^{\text{in}} + (1-\alpha)\,\mathbf{p}^{(t)}\mathbf{T}$, and we set $\mathbf{p}^{(0)} = \mathbf{d}^{\text{in}}$. The graph's flow matrix $\mathbf{F}_{n\times n}$ encodes the flow between each pair of nodes, where $\mathbf{F} = \frac{\alpha}{w_{\text{tot}}}\mathbf{A} + (1-\alpha)\,\text{diag}\,(\mathbf{p})\,\mathbf{T}$. We obtain the flow $\mathbf{C}_{s\times s}$ between clusters from $\mathbf{S}$ and $\mathbf{F}$ as $\mathbf{C} = \mathbf{S}^\top\mathbf{F}\mathbf{S}$. Following Equation (2), we define

$$q = 1 - \text{tr}\,(\mathbf{C}) \qquad \mathbf{q}_{\mathsf{m}} = \mathbf{C}\mathbf{1}_s - \text{diag}\,(\mathbf{C}) \qquad \mathbf{m}_{\text{exit}} = \left(\mathbf{1}_s^\top\mathbf{C}\right)^\top - \text{diag}\,(\mathbf{C}) \qquad \mathbf{p}_{\mathsf{m}} = \mathbf{q}_{\mathsf{m}} + \mathbf{1}_s^\top\mathbf{C}$$

and assemble the map equation

$$L\,(\mathbf{A},\mathbf{S}) = q\log_2 q - (\mathbf{q}_{\mathsf{m}}\log_2\mathbf{q}_{\mathsf{m}})\,\mathbf{1}_s - (\mathbf{m}_{\text{exit}}\log_2\mathbf{m}_{\text{exit}})\,\mathbf{1}_s - (\mathbf{p}\log_2\mathbf{p})\,\mathbf{1}_n + (\mathbf{p}_{\mathsf{m}}\log_2\mathbf{p}_{\mathsf{m}})\,\mathbf{1}_s \quad (3)$$

where $\mathbf{1}_k$ is the k-dimensional vector of ones, and logarithms are applied component-wise. The third term is constant since it only depends on the node visit rates and can be omitted during optimisation.

The map equation naturally incorporates Occam's razor by following the MDL principle for balancing between model complexity and fit [14, 15], choosing the optimal number of communities automatically, but at most s. In contrast, recent GNN-based clustering approaches require explicit regularisation to avoid over-partitioning [10, 36, 37, 38], and our results show that they often return the maximum allowed number of communities instead of determining the number of communities in a data-driven fashion (see Section 5). In principle, any neural network architecture, such as a multi-layer perceptron (MLP) or GNN, can be used to learn the soft cluster assignment matrix $\mathbf{S}$. Since the map equation involves logarithms, we add a small constant ϵ to each value in the output $\mathbf{S}$ before the backpropagation step to ensure differentiability. We refer to the combination of using map equation loss (Equation (3)) together with a (graph) neural network to learn (overlapping) communities as *Neuromap*.

Complexity and limitations. The most expensive calculation is the pooling operation $\mathbf{C} = \mathbf{S}^\top\mathbf{F}\mathbf{S}$ which depends on the network's density. When $s \ll n$ and the number of edges is $m = \mathcal{O}\,(n)$, the complexity of Neuromap is linear in n. When the network is dense, $m = \mathcal{O}\left(n^2\right)$, or the maximum number of clusters approaches the number of nodes $s \approx n$, we approach quadratic complexity. Therefore, we recommend keeping $s \ll n$ for scalability.

We assume connected networks, otherwise, clustering should be run on the individual components. The node features $\mathbf{X}$ aid the GNN in learning patterns, however, they do not contribute to the loss. When no node features are available, Neuromap can use, for example, the adjacency matrix as node features; designing expressive low-dimensional node features remains an active research area [53].

5 Experimental evaluation

We evaluate Neuromap on synthetic and real-world networks with different neural network architectures: a 2-layer graph convolutional network (GCN) [9], a 2-layer graph isomorphism network (GIN) [54], and a 2-layer SAGE network [55]. To investigate whether GNNs are required for clustering, we also include a fully connected linear layer and a 2-layer MLP. We include a learnable temperature

https://doi.org/10.52202/079017-0554

parameter for the softmax operation, which we found speeds up convergence. In all cases, we use the models provided by PyTorch Geometric with SELU activation [56]. Because the specifics between architectures differ, such as message-passing details and aggregation functions, they may be interpreted as using different search algorithms which return different communities. We use the Adam optimiser [57], apply batch normalisation, and for comparability between different methods, set the learning rate for the linear layer to 10^{-1}, for MLP to 10^{-2}, and for GCN, GIN, and SAGE to 10^{-3}. We train all models for up to 10,000 epochs with a patience of 100 epochs and dropout probability 0.5. Because the datasets contain hard clusters, we convert the resulting communities to hard clusters, assigning each node to that cluster where it has its strongest membership. As baselines, we use Infomap [6] and five recent approaches for unsupervised graph clustering with GNNs: DMoN [36], NOCD [38], DiffPool [10], MinCut [11], and Ortho [36]. We base our implementation[3] on PyTorch [58] and PyTorch Geometric [59] and ran our experiments on a workstation with an Intel i9-11900K @ 3.50GHz CPU, 32 GB of RAM, and a GeForce RTX 3090 with 24 GB of memory.

5.1 Synthetic networks with planted communties

We generate directed and undirected Lancichinetti-Fortunato-Radicchi (LFR) benchmark networks with planted ground-truth communities [60] with $n = 1000$ nodes, average node degree $k \in \{\ln n, 2 \ln n\}$, maximum node degree $k_{\max} = 2\sqrt{n}$, both rounded to the nearest integer, and mixing parameter μ between 0.1 and 0.8 with a step size of 0.1. We set the power-law exponents for the node degree distribution to $\tau_1 = 2$, and for the community size distribution to $\tau_2 = 1$. For each combination of parameters, we generate 10 LFR networks using the implementation provided by the authors,[4] resulting in a total of 320 networks. For each parameter combination, there are 10 LFR networks; for each of these LFR networks, we run each model 10 times, measuring its performance as the average adjusted mutual information (AMI) [61] against the ground truth, and plot the average of those AMI values over the 10 networks per parameter combination. To verify that the number of communities is inferred from the data, we set the maximum number of communities to $s = n$. Since LFR networks do not have node features, we use the adjacency matrix as node features.

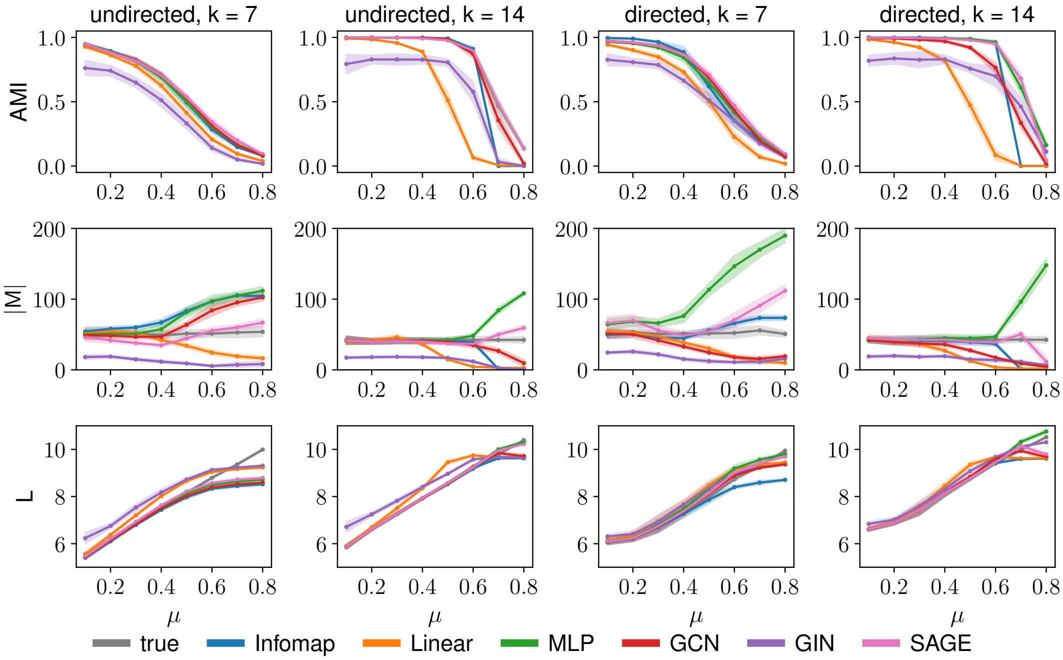


Figure 3: Performance for Neuromap using a dense linear layer, MLP, GCN, GIN, and SAGE architectures with two layers and Infomap on directed and undirected LFR networks with planted communities. The results show averages of partition quality measured with AMI, number of detected communities |M|, and codelength L. The shaded areas show one standard deviation from the mean.

[3] https://github.com/chrisbloecker/neuromap

[4] https://sites.google.com/site/andrealancichinetti/benchmarks

https://doi.org/10.52202/079017-0554

We find that the detected communities' quality depends on the choice of neural network architecture (Figure 3). Neuromap achieves the best AMI scores with SAGE. GCN, MLP, and Infomap perform slightly worse, however, with some variation depending on the networks' properties. The dense linear layer and GIN show weaker performance but still identify relevant communities. In the sparser directed networks, Infomap performs slightly better than SAGE when the mixing μ is low. However, the AMI values do not tell the whole story: Infomap and MLP tend to report considerably more communities than are present in the ground truth whereas the dense linear layer and GIN tend to report much fewer communities than the ground truth, especially for higher mixing values. GCN reports more communities than are present in the ground truth in the sparser undirected networks but fewer in the directed networks. SAGE detects close to the true number of communities in all cases. Infomap achieves the lowest codelength across all networks. GCN, MLP, and SAGE achieve close to Infomap's codelength, whereas the dense linear layer and GIN have slightly higher codelength.

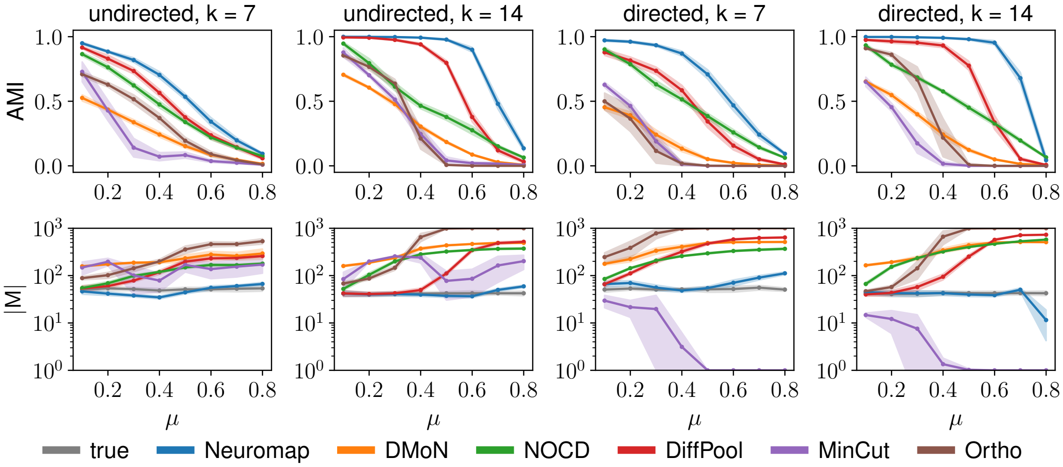


Figure 4: SAGE-based results for deep learning community-detection methods on synthetic LFR networks with planted communities. We show averages of partition quality measured by AMI and number of detected communities |M|. The shaded areas show one standard deviation from the mean.

We compare Neuromap against recent deep-learning-based community detection methods on the same networks by swapping out the loss function while keeping everything else the same, with the exception of using weight decay for NOCD as per the original paper [38]. For DiffPool, Mincut, Ortho, and DMoN, we use the implementation from PyTorch Geometric, for NOCD, we use the implementation provided by the authors.[5] Figure 4 shows the results for SAGE; in Appendix B, we also include results for the remaining architectures. Neuromap outperforms the baselines across all architectures, except for NOCD which performs better than Neuromap with GIN and with GCN on directed networks. Different from previous works, we have not limited the maximum number of communities, which allows us to analyse the methods' overfitting behaviour: While Neuromap reports close to the ground-truth number of communities, the remaining methods often overfit the networks' structure and report considerably more communities (note the logarithmic scale). MinCut fails to identify meaningful communities on directed networks for mixing values $\mu > 0.3$. NOCD performs best with the GCN architecture, which was also used in the original paper [38]. Neuromap performs best with SAGE in our experiments.

5.2 Real-world networks with node features

We benchmark Neuromap on ten real-world datasets (Table 1) from PyTorch Geometric [59], PyTorch Geometric Signed Directed [62], and Open Graph Benchmark [63], and compare it against the same baselines as before. In contrast to previous works that choose a fixed number of hidden dimensions and set the maximum number of communities to a constant [36] or the "ground-truth" number of communities [38], we reflect the networks' sizes in our choices: We set the number of hidden dimensions to $4\sqrt{n}$ and the maximum number of communities to $s = \sqrt{n}$. Our choices are based on empirical observations showing that the number of communities typically scales as $\mathcal{O}\left(\sqrt{n}\right)$ [64].

[5] `https://github.com/shchur/overlapping-community-detection`

Table 1: Properties of the real-world datasets obtained from PyTorch Geometric (PyG) [59], PyTorch Geometric Signed Directed (PyG-SD) [62], and Open Graph Benchmark (OGB) [63]. $|V|$ is the number of nodes, $|E|$ the number of edges, $|X|$ the node feature dimension, $|Y|$ the number of communities, and μ the mixing for the given communities.

Dataset	Source	Type	$\|V\|$	$\|E\|$	$\|X\|$	$\|Y\|$	μ
Cora	PyG	Undirected	2,708	5,278	1,433	7	0.19
CiteSeer	PyG	Undirected	3,327	4,614	3,703	6	0.26
Pubmed	PyG	Undirected	19,717	44,325	500	3	0.20
Amazon Computer (PC)	PyG	Undirected	13,752	143,604	767	10	0.22
Amazon Photo	PyG	Undirected	7,650	71,831	745	8	0.17
Coauthor CS	PyG	Undirected	18,333	81,894	6,805	15	0.19
Coauthor Physics	PyG	Undirected	34,493	247,962	8,415	5	0.07
Cora ML	PyG-SD	Directed	2,995	8,416	2,879	7	0.21
Wiki CS	PyG-SD	Directed	11,701	297,110	300	10	0.31
ogb-arxiv	OGB	Directed	169,343	583,121	128	40	0.35

However, a few words of caution are in order: while nodes' true communities determine the link patterns in synthetic networks, it is generally infeasible to obtain ground truth communities for real networks. Often, metadata labels are used as a drop-in, and the inferred communities' quality depends on how well the metadata, which is potentially noisy, aligns with the unknown ground truth [3]. Moreover, determining the number of communities in a network is hard and setting $s = \sqrt{n}$ should be seen as a simplification rather than an attempt to guess the exact number.

For each method and architecture, we run 25 trials and show the average achieved AMI in Figure 5; we include a similar plot for the number of detected communities as well as the average AMI and the detected number of communities, both with standard deviations, in tabulated form in Appendix C. When several of the best-performing methods achieve similar AMI, we use an independent two-sample t-test to determine whether one of them can be considered to perform better than the other (see Appendix C). In cases where their performances do not differ significantly, we mark both as best.

Neuromap and NOCD are amongst the best performers in seven cases and DiffPool in two. The GCN architecture performs best in seven cases, MLP and SAGE in four cases, and GIN in two cases. A

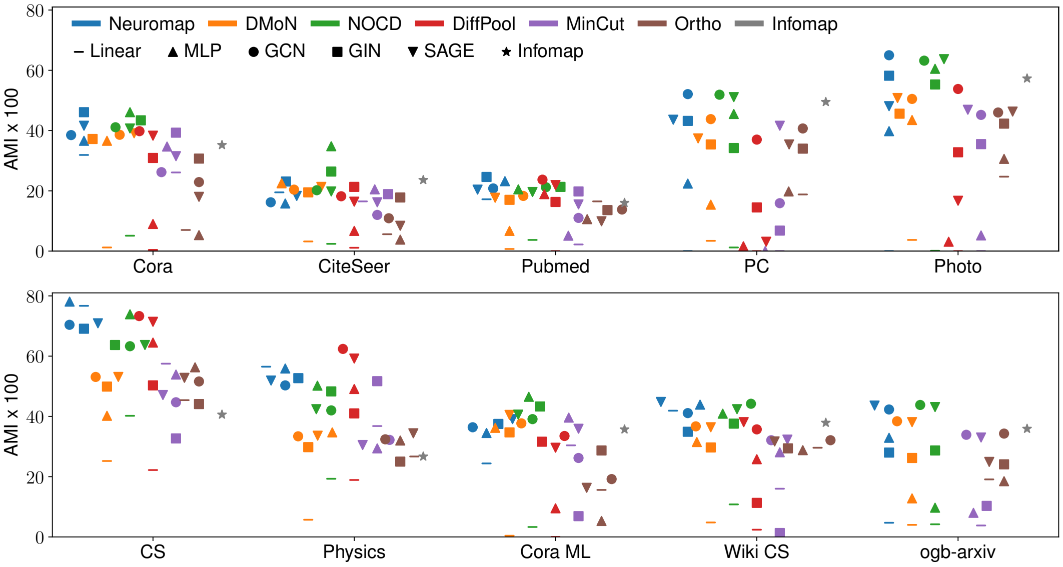


Figure 5: Average achieved AMI on real-world networks (higher is better) with $s = \sqrt{n}$. Colours indicate methods while shapes indicate neural network architectures. DiffPool ran out of memory on the ogb-arxiv dataset. Detailed tabulated results with standard deviations are included in Appendix C.

https://doi.org/10.52202/079017-0554

possible explanation for why the simpler linear layer and MLP architectures perform well in several cases could be that the map equation loss function captures global information in the random walker's flow patterns, making GNNs superfluous in some cases. All methods tend to detect more communities than are present in the "ground truth", however, this tendency is most pronounced in DMoN, Ortho, and MinCut, which may have gone unnoticed in previous evaluations where the maximum number of communities was set to a much lower, constant value [36], thus artificially preventing overfitting. Infomap is the only baseline that does not utilise node features; instead, it relies solely on topological information, which may explain the large number of detected communities. Comparing Neuromap's performance against Infomap's performance suggests that incorporating node features substantially improves the detected communities' quality in most cases (see Appendix C for significance tests) while drastically reducing the number of detected communities.

We repeat the same experiments with 512 hidden features and $s = |Y|$, that is, the "true" number of communities, following [38] (results in Appendix D). Limiting the number of allowed communities often leads to better performance for DMoN, MinCut, and Ortho, however, with a few exceptions, it diminishes the performance of Neuromap, NOCD, and DiffPool across all datasets and neural network architectures. Appendix E tabulates the differences in average AMI score between setting the hidden features to $4\sqrt{n}$ and $s = \sqrt{n}$ versus using 512 hidden features and $s = |Y|$. In the case of Neuromap, imposing a lower bound on the number of communities interferes with the MDL principle, limiting what models for the data may be explored.

5.3 Synthetic networks with overlapping communities

We apply Neuromap and the baselines to a small synthetic network with overlapping communities [65]. We set the maximum number of communities to $s \in \{2, 3\}$, run each combination of loss function and neural network architecture for 10 trials, and keep the solutions with the lowest loss. Figure 6 shows the results obtained with GCN, Appendix F shows results for the remaining architectures.

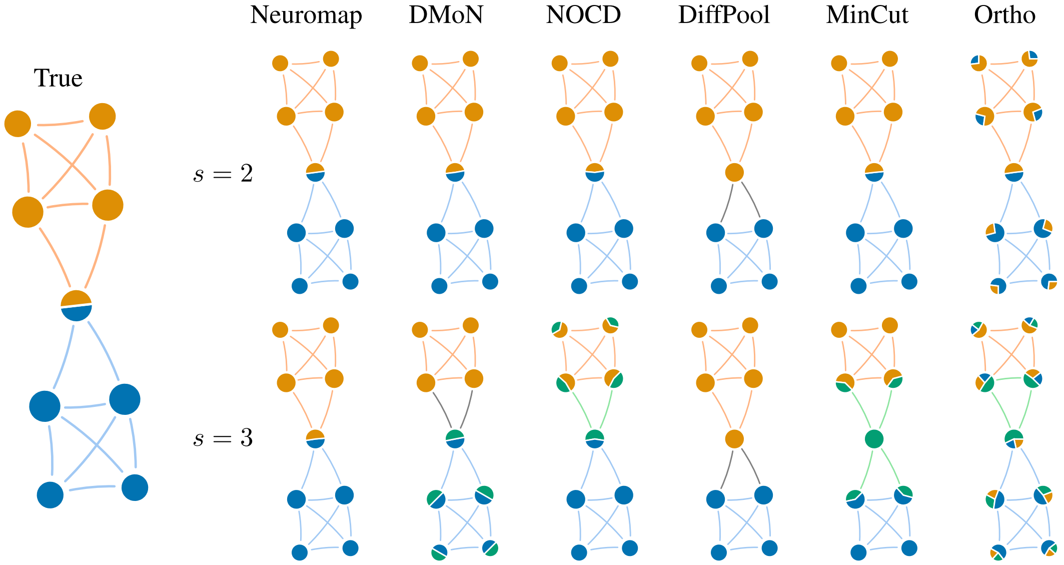


Figure 6: Synthetic network with overlapping communities where the leftmost network shows the true community structure. Nodes are drawn as pie charts to visualise their community assignments. The top and bottom rows show results for a maximum of $s = 2$ and $s = 3$ communities, respectively.

We find that Neuromap, DMoN, NOCD, and MinCut identify the correct communities for $s = 2$. DiffPool does not detect overlapping communities and Ortho assigns each node to two communities. For $s = 3$, only Neuromap identifies the correct communities. DiffPool returns the same communities as for $s = 2$. All remaining methods return three communities. These results provide further evidence that the baselines suffer from overfitting when they are not provided with the correct number of communities, which, in general, is unknown. Neuromap identifies meaningful communities while inferring the number of communities in a data-driven fashion by following the MDL principle. However, we leave a more rigorous study of overlapping communities for future work.

https://doi.org/10.52202/079017-0554

6 Conclusion

Network science and deep learning on graphs tackle community detection from different perspectives. Community detection in network science typically relies on custom heuristic optimisation algorithms to optimise objective functions but often does not leverage recent deep learning advances. Recently, deep graph learning methods have started to incorporate methods from network science for deep graph clustering. We contribute to this young field by adapting the map equation, a popular unsupervised information-theoretic community-detection approach, as a differentiable loss function for end-to-end optimisation with (graph) neural networks through gradient descent, and use PyTorch to implement our approach, which we call Neuromap.

We evaluated Neuromap on various synthetic and real-world datasets, using different neural network architectures to detect communities. Our results show that Neuromap achieves competitive performance and detects close to the ground-truth number of communities across datasets while the baselines tend to overfit and report considerably more communities. Across all tested methods, the achieved performance depends on the used neural network architecture. However, on several real-world benchmarks, Neuromap outperforms several of the the baselines even with simpler, non-GNN, neural network architectures. We hypothesise that this may be because the map equation builds on capturing flow patterns, which contain global information.

While we have considered first-order networks with two-level community structures, complex real-world networks often involve higher-order dependencies and can have multi-level communities [47, 66], prompting a generalisation of our approach. Furthermore, incorporating our method for graph pooling as well as uncovering the precise connection between the utilised neural network architecture and the achieved community-detection performance requires further empirical and theoretical studies.

Acknowledgments and Disclosure of Funding

We thank Martin Rosvall, Jelena Smiljanić, and Daniel Edler for fruitful discussions and helpful feedback. We are indebted to Lisi Qarkaxhija for helping with PyTorch implementations. Christopher Blöcker and Ingo Scholtes acknowledge funding from the Swiss National Science Foundation, grant 176938, and the German Federal Ministry of Education and Research, grant 100582863 (TissueNet).

References

[1] Santo Fortunato. Community detection in graphs. *Physics Reports*, 486(3):75–174, 2010.

[2] Santo Fortunato and Mark EJ Newman. 20 years of network community detection. *Nature Physics*, 18(8):848–850, 2022.

[3] Leto Peel, Daniel B. Larremore, and Aaron Clauset. The ground truth about metadata and community detection in networks. *Science Advances*, 3(5):e1602548, 2017.

[4] Vincent D Blondel, Jean-Loup Guillaume, Renaud Lambiotte, and Etienne Lefebvre. Fast unfolding of communities in large networks. *Journal of Statistical Mechanics: Theory and Experiment*, 2008(10):P10008, oct 2008.

[5] V. A. Traag, L. Waltman, and N. J. van Eck. From louvain to leiden: guaranteeing well-connected communities. *Scientific Reports*, 9(1):5233, Mar 2019.

[6] Daniel Edler, Ludvig Bohlin, and Martin Rosvall. Mapping higher-order network flows in memory and multilayer networks with infomap. *Algorithms*, 10(4), 2017.

[7] 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, 2009.

[8] Justin Gilmer, Samuel S. Schoenholz, Patrick F. Riley, Oriol Vinyals, and George E. Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 1263–1272. PMLR, 06–11 Aug 2017.

[9] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations*, 2017.

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

[11] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In *International conference on machine learning*, pages 874–883. PMLR, 2020.

[12] Filippo Maria Bianchi and Veronica Lachi. The expressive power of pooling in graph neural networks. In *Advances in Neural Information Processing Systems*, volume 36, pages 71603–71618. Curran Associates, Inc., 2023.

[13] Satu Elisa Schaeffer. Graph clustering. *Computer science review*, 1(1):27–64, 2007.

[14] Jorma Rissanen. Modeling by shortest data description. *Automatica*, 14(5):465–471, 1978.

[15] Peter D Grünwald, In Jae Myung, and Mark A Pitt. *Advances in minimum description length: Theory and applications*. MIT press, 2005.

[16] Martin Rosvall and Carl T. Bergstrom. Maps of random walks on complex networks reveal community structure. *Proceedings of the National Academy of Sciences*, 105(4):1118–1123, 2008.

[17] M. E. J. Newman. Modularity and community structure in networks. *Proceedings of the National Academy of Sciences*, 103(23):8577–8582, 2006.

[18] Paul W. Holland, Kathryn Blackmond Laskey, and Samuel Leinhardt. Stochastic blockmodels: First steps. *Social Networks*, 5(2):109–137, 1983.

[19] Tiago P Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 89(1):012804, 2014.

[20] M. Rosvall, D. Axelsson, and C. T. Bergstrom. The map equation. *The European Physical Journal Special Topics*, 178(1):13–23, Nov 2009.

[21] Geoffrey E Hinton and Drew Van Camp. Keeping the neural networks simple by minimizing the description length of the weights. In *Proceedings of the sixth annual conference on Computational learning theory*, pages 5–13, 1993.

[22] Léonard Blier and Yann Ollivier. The description length of deep learning models. In *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018.

[23] Ethan Perez, Douwe Kiela, and Kyunghyun Cho. Rissanen data analysis: Examining dataset characteristics via description length. In *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8500–8513. PMLR, 18–24 Jul 2021.

[24] Tiago P. Peixoto. Hierarchical block structures and high-resolution model selection in large networks. *Phys. Rev. X*, 4:011047, Mar 2014.

[25] Kai Yu, Shipeng Yu, and Volker Tresp. Soft clustering on graphs. *Advances in neural information processing systems*, 18, 2005.

[26] Jianbo Shi and Jitendra Malik. Normalized cuts and image segmentation. *IEEE Transactions on pattern analysis and machine intelligence*, 22(8):888–905, 2000.

[27] Ulrike von Luxburg. A tutorial on spectral clustering. *Statistics and Computing*, 17(4):395–416, Dec 2007.

[28] Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '14, page 701–710, New York, NY, USA, 2014. Association for Computing Machinery.

https://doi.org/10.52202/079017-0554

[29] Aditya Grover and Jure Leskovec. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, page 855–864, New York, NY, USA, 2016. Association for Computing Machinery.

[30] Aditya Tandon, Aiiad Albeshri, Vijey Thayananthan, Wadee Alhalabi, Filippo Radicchi, and Santo Fortunato. Community detection in networks using graph embeddings. *Physical Review E*, 103(2):022316, 2021.

[31] Sadamori Kojaku, Filippo Radicchi, Yong-Yeol Ahn, and Santo Fortunato. Network community detection via neural embeddings. *arXiv preprint arXiv:2306.13400*, 2023.

[32] Chun Wang, Shirui Pan, Guodong Long, Xingquan Zhu, and Jing Jiang. Mgae: Marginalized graph autoencoder for graph clustering. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 889–898, 2017.

[33] Nairouz Mrabah, Mohamed Bouguessa, Mohamed Fawzi Touati, and Riadh Ksantini. Rethinking graph auto-encoder models for attributed graph clustering. *IEEE Transactions on Knowledge and Data Engineering*, 2022.

[34] Maedeh Ahmadi, Mehran Safayani, and Abdolreza Mirzaei. Deep graph clustering via mutual information maximization and mixture model. *Knowledge and Information Systems*, Apr 2024.

[35] Sambaran Bandyopadhyay and Vishal Peter. Unsupervised constrained community detection via self-expressive graph neural network. In *Uncertainty in Artificial Intelligence*, pages 1078–1088. PMLR, 2021.

[36] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *Journal of Machine Learning Research*, 24(127):1–21, 2023.

[37] Tsuyoshi Murata and Naveed Afzal. Modularity optimization as a training criterion for graph neural networks. In *Complex Networks IX*, pages 123–135, Cham, 2018. Springer International Publishing.

[38] Oleksandr Shchur and Stephan Günnemann. Overlapping community detection with graph neural networks. *Deep Learning on Graphs Workshop, KDD*, 2019.

[39] Liu Yue, Xia Jun, Zhou Sihang, Wang Siwei, Guo Xifeng, Yang Xihong, Liang Ke, Tu Wenxuan, Liu Xin Wang, et al. A survey of deep graph clustering: Taxonomy, challenge, and application. *arXiv preprint arXiv:2211.12875*, 2022.

[40] Su Xing, Xue Shan, Liu Fanzhen, Wu Jia, Yang Jian, Zhou Chuan, Hu Wenbin, Paris Cecile, Nepal Surya, Jin Di, et al. A comprehensive survey on community detection with deep learning. *IEEE Trans. Neural Netw. Learn. Syst*, 2022.

[41] Andrea Lancichinetti and Santo Fortunato. Community detection algorithms: A comparative analysis. *Phys. Rev. E*, 80:056117, 2009.

[42] Rodrigo Aldecoa and Ignacio Marín. Exploring the limits of community detection strategies in complex networks. *Sci. Rep.*, 3:2216, 2013.

[43] Lovro Šubelj, Nees Jan van Eck, and Ludo Waltman. Clustering scientific publications based on citation relations: A systematic comparison of different methods. *PLOS ONE*, 11(4):1–23, 04 2016.

[44] C. E. Shannon. A mathematical theory of communication. *Bell Syst. Tech. J.*, 27:379–423, 1948.

[45] R. Lambiotte and M. Rosvall. Ranking and clustering of nodes in networks with smart teleportation. *Phys. Rev. E*, 85:056107, May 2012.

[46] David A. Huffman. A method for the construction of minimum-redundancy codes. *Proceedings of the IRE*, 40(9):1098–1101, 1952.

[47] Martin Rosvall, Alcides V. Esquivel, Andrea Lancichinetti, Jevin D. West, and Renaud Lambiotte. Memory in network flows and its effects on spreading dynamics and community detection. *Nature Communications*, 5(1):4630, Aug 2014.

[48] Anton Holmgren, Christopher Blöcker, and Martin Rosvall. Mapping biased higher-order walks reveals overlapping communities, 2023.

[49] Jelena Smiljanić, Christopher Blöcker, Daniel Edler, and Martin Rosvall. Mapping flows on weighted and directed networks with incomplete observations. *Journal of Complex Networks*, 9(6):cnab044, 12 2021.

[50] Daniel Edler, Jelena Smiljanić, Anton Holmgren, Alexandre Antonelli, and Martin Rosvall. Variable markov dynamics as a multi-focal lens to map multi-scale complex networks, 2022.

[51] Scott Emmons and Peter J. Mucha. Map equation with metadata: Varying the role of attributes in community detection. *Phys. Rev. E*, 100:022301, Aug 2019.

[52] Aleix Bassolas, Anton Holmgren, Antoine Marot, Martin Rosvall, and Vincenzo Nicosia. Mapping nonlocal relationships between metadata and network structure with metadata-dependent encoding of random walks. *Science Advances*, 8(43):eabn7558, 2022.

[53] Derek Lim, Joshua David Robinson, Lingxiao Zhao, Tess Smidt, Suvrit Sra, Haggai Maron, and Stefanie Jegelka. Sign and basis invariant networks for spectral graph representation learning. In *The Eleventh International Conference on Learning Representations*, 2023.

[54] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *International Conference on Learning Representations*, 2019.

[55] Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017.

[56] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. *Advances in neural information processing systems*, 30, 2017.

[57] Diederik P. Kingma and Jimmy Ba. Adam: A method for stochastic optimization, 2017.

[58] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. PyTorch: An Imperative Style, High-Performance Deep Learning Library. In *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

[59] Matthias Fey and Jan Eric Lenssen. Fast graph representation learning with pytorch geometric. *arXiv preprint arXiv:1903.02428*, 2019.

[60] Andrea Lancichinetti, Santo Fortunato, and Filippo Radicchi. Benchmark graphs for testing community detection algorithms. *Phys. Rev. E*, 78:046110, Oct 2008.

[61] Simone Romano, James Bailey, Vinh Nguyen, and Karin Verspoor. Standardized mutual information for clustering comparisons: One step further in adjustment for chance. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, pages 1143–1151, Bejing, China, 22–24 Jun 2014. PMLR.

[62] Yixuan He, Xitong Zhang, Junjie Huang, Benedek Rozemberczki, Mihai Cucuringu, and Gesine Reinert. Pytorch geometric signed directed: A software package on graph neural networks for signed and directed graphs. In *Learning on Graphs Conference*, pages 12–1. PMLR, 2024.

[63] Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. *Advances in neural information processing systems*, 33:22118–22133, 2020.

https://doi.org/10.52202/079017-0554

[64] Amir Ghasemian, Homa Hosseinmardi, and Aaron Clauset. Evaluating overfit and underfit in models of network community structure. *IEEE Transactions on Knowledge and Data Engineering*, 32(9):1722–1735, 2020.

[65] Alcides Viamontes Esquivel and Martin Rosvall. Compression of flow can reveal overlapping-module organization in networks. *Phys. Rev. X*, 1:021025, Dec 2011.

[66] M. Rosvall and C. T. Bergstrom. Multilevel Compression of Random Walks on Networks Reveals Hierarchical Organization in Large Integrated Systems. *PLoS One*, 6:e18209, 2011.

[67] David F. Gleich. Pagerank beyond the web. *SIAM Review*, 57(3):321–363, 2015.

https://doi.org/10.52202/079017-0554

A Map equation details

Let $G = (V, E)$ be a graph with nodes V, links E, and let $w_{uv} \in \mathbb{R}_0^+$ denote the non-negative weight on the link from node u to v. Further, let p_u be node u's ergodic visit rate, which can be calculated in closed form as $p_u = s_u / \sum_{v \in V} s_v$ in undirected graphs, where $s_u = \sum_{v \in V} w_{uv}$ is node u's strength. In directed graphs, p_u can be calculated numerically with smart teleportation and a power iteration by solving the recursive set of equations $p_v = \alpha \frac{s_v^{\text{out}}}{\sum_{u \in V} s_u^{\text{out}}} + (1 - \alpha) \sum_{u \in V} p_u t_{uv}$, where $s_u^{\text{out}} = \sum_{v \in V} w_{uv}$ is node u's out-strength, $t_{uv} = w_{uv} / \sum_{v \in V} w_{uv}$ is the random walker's transition probability from node u to node v, and α is a teleportation parameter, typically set to $\alpha = 0.15$ [45]. Smart teleportation is similar to PageRank [67] but, instead of uniformly teleporting to nodes, the random walker teleports to links proportional to their weight.

For a given partition M of the nodes into modules, the map equation [16] calculates the average per-step description length – also called codelength – for describing the position of a random walker on the graph:

$$L(\mathsf{M}) = qH(Q) + \sum_{\mathsf{m} \in \mathsf{M}} p_{\mathsf{m}} H(P_{\mathsf{m}}), \tag{4}$$

Here, $q = \sum_{\mathsf{m}} q_{\mathsf{m}}$ is the random walker's module entry rate, $q_{\mathsf{m}} = \sum_{u \notin \mathsf{m}} \sum_{v \in \mathsf{m}} p_u t_{uv}$ is module m's entry rate, and $Q = \{q_{\mathsf{m}}/q \mid \mathsf{m} \in \mathsf{M}\}$ is the set of normalised module entry rates; $p_{\mathsf{m}} = \mathsf{m}_{\text{exit}} + \sum_{u \in \mathsf{m}} p_u$ is the rate at which the random walker moves in module m, including the exit rate $\mathsf{m}_{\text{exit}} = \sum_{u \in \mathsf{m}} \sum_{v \notin \mathsf{m}} p_u t_{uv}$, and $P_{\mathsf{m}} = \{\mathsf{m}_{\text{exit}}/p_{\mathsf{m}}\} \cup \{p_u/p_{\mathsf{m}} \mid u \in \mathsf{m}\}$ is the set of normalised node visit rates and exit rates for module m.

A.1 Rewriting the map equation

The map equation can be rewritten by expanding the entropy terms and cancelling common factors

$$L(\mathsf{M}) = qH(Q) + \sum_{\mathsf{m} \in \mathsf{M}} p_{\mathsf{m}} H(P_{\mathsf{m}}) \tag{5}$$

$$= -\cancel{q} \sum_{\mathsf{m} \in \mathsf{M}} \frac{q_{\mathsf{m}}}{\cancel{q}} \log_2 \frac{q_{\mathsf{m}}}{q} - \sum_{\mathsf{m} \in \mathsf{M}} \cancel{p_{\mathsf{m}}} \left(\frac{\mathsf{m}_{\text{exit}}}{\cancel{p_{\mathsf{m}}}} \log_2 \frac{\mathsf{m}_{\text{exit}}}{p_{\mathsf{m}}} + \sum_{u \in \mathsf{m}} \frac{p_u}{\cancel{p_{\mathsf{m}}}} \log_2 \frac{p_u}{p_{\mathsf{m}}} \right). \tag{6}$$

Applying logarithm rules gives

$$= -\sum_{\mathsf{m} \in \mathsf{M}} q_{\mathsf{m}} \log_2 q_{\mathsf{m}} + \sum_{\mathsf{m} \in \mathsf{M}} q_{\mathsf{m}} \log_2 q \tag{7}$$

$$- \sum_{\mathsf{m} \in \mathsf{M}} \mathsf{m}_{\text{exit}} \log_2 \mathsf{m}_{\text{exit}} + \sum_{\mathsf{m} \in \mathsf{M}} \mathsf{m}_{\text{exit}} \log_2 p_{\mathsf{m}} - \sum_{\mathsf{m} \in \mathsf{M}} \sum_{u \in \mathsf{m}} p_u \log_2 p_u + \sum_{\mathsf{m} \in \mathsf{M}} \sum_{u \in \mathsf{m}} p_u \log_2 p_{\mathsf{m}}, \tag{8}$$

and further simplification yields

$$= q \log_2 q - \sum_{\mathsf{m} \in \mathsf{M}} q_{\mathsf{m}} \log_2 q_{\mathsf{m}} + \sum_{\mathsf{m} \in \mathsf{M}} p_{\mathsf{m}} \log_2 p_{\mathsf{m}} - \sum_{\mathsf{m} \in \mathsf{M}} \mathsf{m}_{\text{exit}} \log_2 \mathsf{m}_{\text{exit}} - \sum_{u \in V} p_u \log_2 p_u. \tag{9}$$

In undirected networks where $q_{\mathsf{m}} = \mathsf{m}_{\text{exit}}$, this can be further simplified [20] to

$$= q \log_2 q - 2 \sum_{\mathsf{m} \in \mathsf{M}} q_{\mathsf{m}} \log_2 q_{\mathsf{m}} + \sum_{\mathsf{m} \in \mathsf{M}} p_{\mathsf{m}} \log_2 p_{\mathsf{m}} - \sum_{u \in V} p_u \log_2 p_u. \tag{10}$$

The last term in Equation (10), that is, the nodes' contribution to the codelength, is constant because it does not depend on the module structure, and can be omitted during optimisation.

A.2 Node flow with soft cluster assignments

While the nodes' codelength contribution is constant with hard clusters (Equation (10)), expressing the map equation with soft cluster assignments allows modelling the flow of nodes that are assigned to more than one module in at least two different ways which we illustrate in Figure 7.

The first option is to reflect the nodes' partial cluster assignment for assigning codewords. Consider node u with visit rate p_u and partial assignments of $\frac{1}{2}$ to clusters m_1 whose usage rate is p_{m_1} and $\frac{1}{2}$

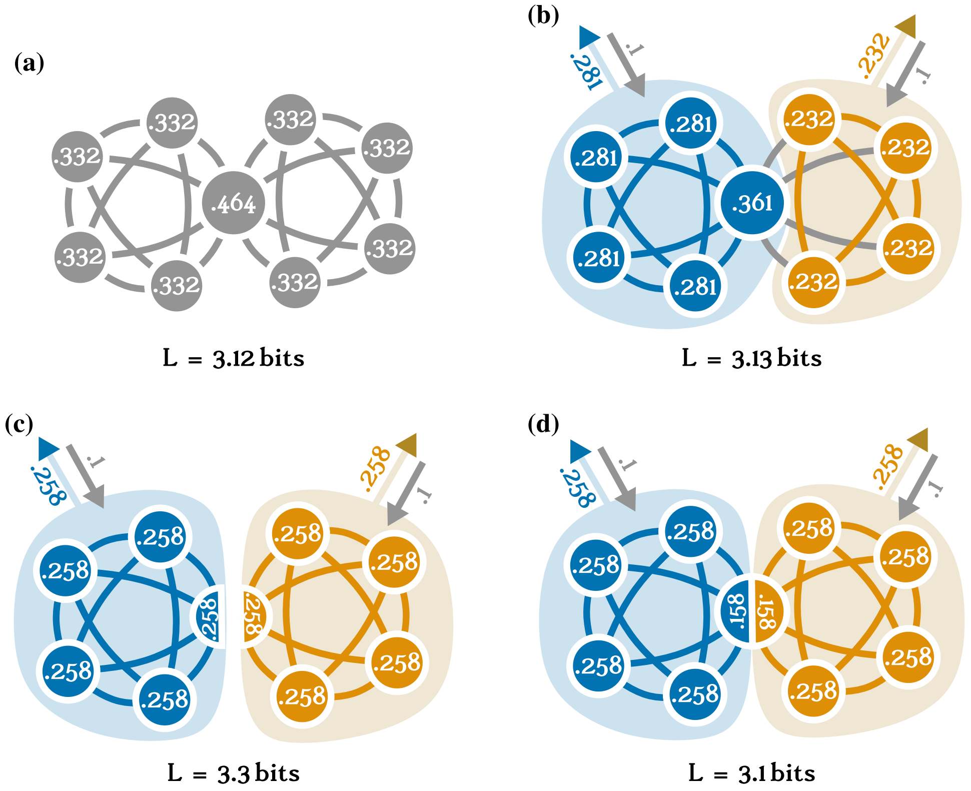


Figure 7: **Modelling node flow with soft cluster assignments.** The labels show each node's contribution to the overall codelength based on their visit rates. **(a)** The one-level partition where all nodes belong to the same community. **(b)** With hard communities, the middle node must be assigned to either the blue or orange community. Here, this leads to a higher codelength than for the one-level partition. **(c)** Reflecting nodes' partial assignments in the codelength contributions means splitting them into several smaller pieces whose visit rates sum to the original node's visit rate. Important objects are split into several less important objects, requiring longer codewords because of their lower visit rates. Overall, this tends to increase the codelength and can prevent identifying overlapping modules. **(d)** Treating nodes' contribution to the codelength as constant keeps the nodes intact and allows important nodes to retain their higher visit rates, leading to shorter codewords. As the codelength highlights, only this approach would identify communities in this example.

to cluster m_2 whose usage rate is p_{m_2}. Then, half of u's flow, that is $\frac{p_u}{2}$, falls into each of the two clusters and u's contribution to the overall codelength is $-\frac{p_u}{2}\log_2\frac{p_u}{2p_{\mathsf{m}_1}} - \frac{p_u}{2}\log_2\frac{p_u}{p_{2\mathsf{m}_2}}$. In general, let node u's assignment to cluster m_i be $s_{ui} \in [0,1]$ with $\sum_i s_{ui} = 1$, and let m_i's usage rate be p_{m_i}. Then, node u's contribution to the overall codelength is $-\sum_i s_{ui} p_u \log_2 \frac{s_{ui} p_u}{p_{\mathsf{m}_i}}$. Essentially, this approach splits each node with assignments to more than one cluster into its corresponding parts, resulting in potentially many small pieces with low visit rates.

The second option is to treat the nodes as indivisible when computing their codeword lengths but reflecting their soft assignments in the contribution to the overall codelength. Using the same example as above, node u's contribution to the overall codelength is $-\frac{p_u}{2}\log_2\frac{p_u}{p_{\mathsf{m}_1}} - \frac{p_u}{2}\log_2\frac{p_u}{p_{\mathsf{m}_2}}$, which is lower because the values inside the logarithms are bigger. In general, u's contribution to the codelength is $-\sum_i s_{ui} p_u \log_2 \frac{p_u}{p_{\mathsf{m}_i}}$, where the difference to the previous option is that s_{ui} does not appear inside the logarithm.

B Results on synthetic networks with different (G)NN architectures

Here, we report further results on the synthetic LFR networks with planted communities for Neuromap, DMoN, NOCD, DiffPool, MinCut, and Ortho using the following architectures: a dense linear layer, a 2-layer MLP, a 2-layer GCN, and a 2-layer GIN. The setup is as described in the main text.

B.1 Linear-layer-based results

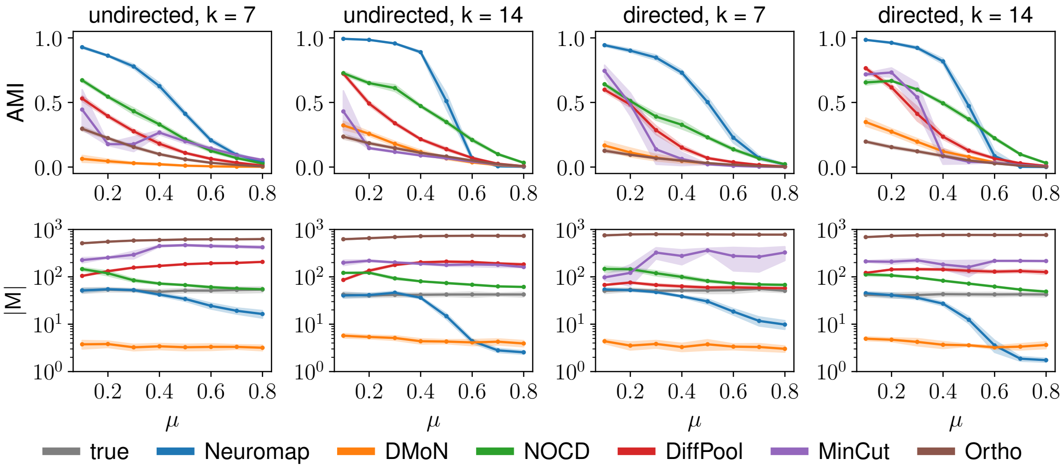


Figure 8: Linear-layer-based results for deep-learning community-detection methods on synthetic LFR networks with planted communities. The results show averages of partition quality measured by AMI and number of detected communities |M|. The shaded areas show one standard deviation from the mean.

B.2 MLP-based results

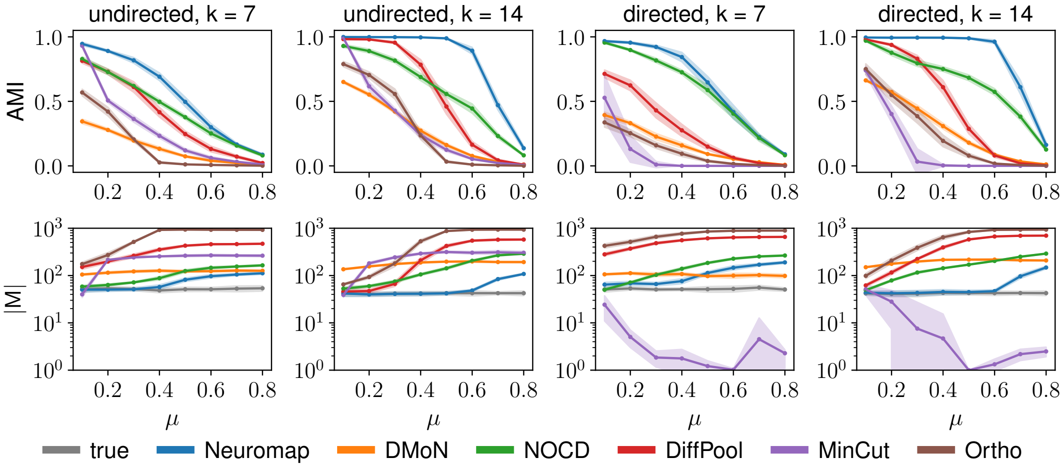


Figure 9: MLP-based results for deep-learning community-detection methods on synthetic LFR networks with planted communities. The results show averages of partition quality measured by AMI and number of detected communities |M|. The shaded areas show one standard deviation from the mean.

B.3 GCN-based results

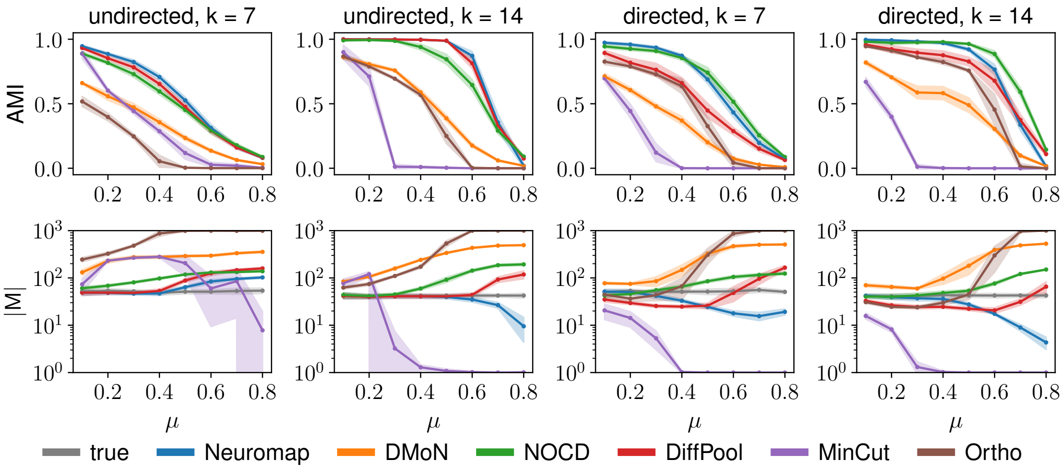


Figure 10: GCN-based results for deep-learning community-detection methods on synthetic LFR networks with planted communities. The results show averages of partition quality measured by AMI and number of detected communities |M|. The shaded areas show one standard deviation from the mean.

B.4 GIN-based results

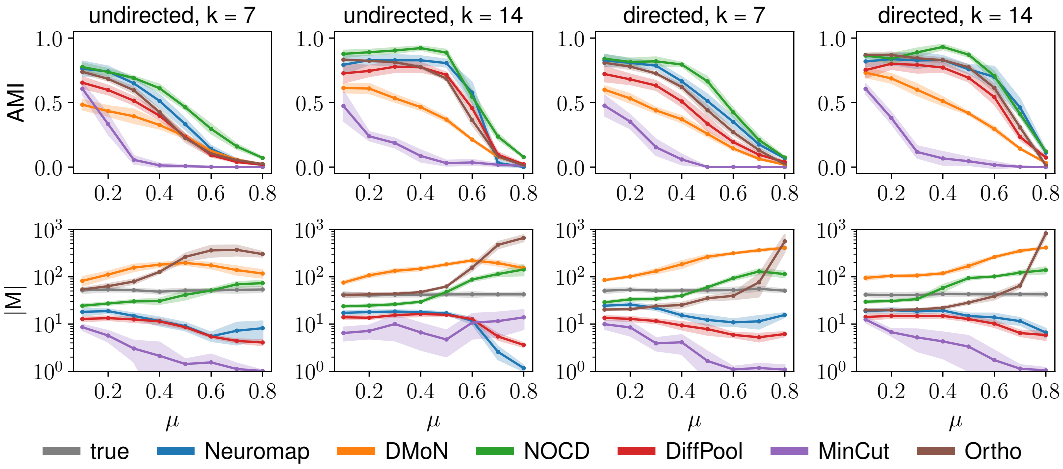


Figure 11: GIN-based results for deep-learning community-detection methods on synthetic LFR networks with planted communities. The results show averages of partition quality measured by AMI and number of detected communities |M|. The shaded areas show one standard deviation from the mean.

https://doi.org/10.52202/079017-0554

C Results on Real-World Networks

Here, we report further results on real-world networks for setting the number of hidden dimensions to $4\sqrt{n}$ and the maximum number of communities to $s = \sqrt{n}$. Specifically, we visualise the average detected number of communities per loss function and neural network architecture for each dataset. We tabulate the average AMI and average number of detected communities, including standard deviations. Two-sample t-tests show when Neuromap or a baseline performs significantly better than the other.

C.1 Number of detected communities on real-world networks

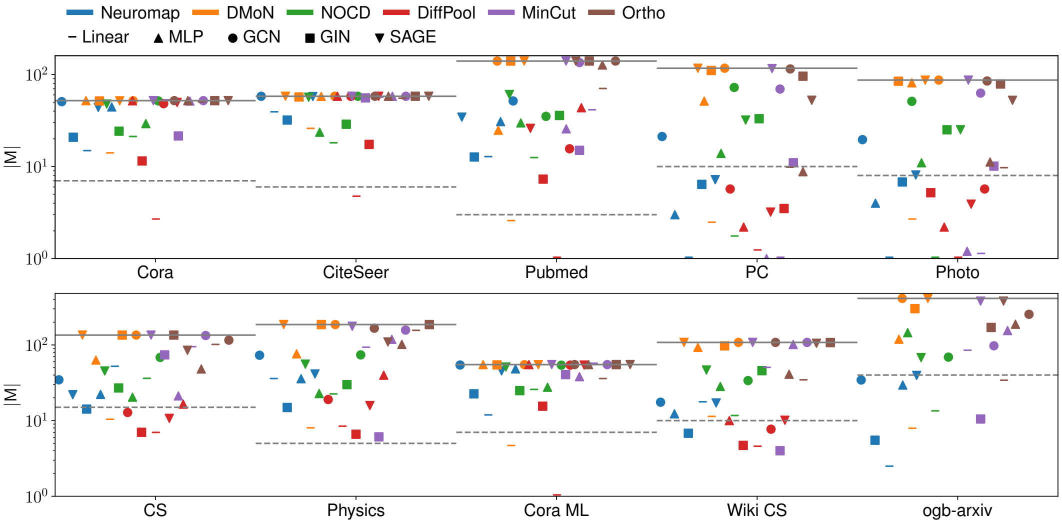


Figure 12: Average detected number of communities on real-world networks. Colours indicate methods while shapes indicate (G)NN architectures. The dashed horizontal lines show the correct number of communities for each dataset while the solid horizontal lines show the chosen maximum allowed number of communities, here $s = \sqrt{n}$. DiffPool ran out of memory on the ogb-arxiv dataset. We omit Infomap in the plot due to the large number of detected communities.

https://doi.org/10.52202/079017-0554

C.2 Tabulated AMI results on real-world networks

Here, we report the detailed AMI performance for Neuromap, DMoN, NOCD, DiffPool, MinCut, Ortho, and Infomap on real-world networks for setting the number of hidden dimensions to $4\sqrt{n}$ and the maximum number of communities to $s = \sqrt{n}$. When several of the best-performing methods achieve similar AMI, we use an independent two-sample t-test to determine whether one of them can be considered to perform better than the other. In cases where their performances do not differ significantly, we mark both as best.

Table 2: Average AMI in % (higher is better) and their standard deviations on real-world networks. For each dataset, we highlight the best scores in bold and underline the second-best score. We tested each method with 2-layer MLP, GCN, GIN, and SAGE architectures, except for Infomap, which is not based on deep learning. OOM stands for "out of memory".

Method	Arch.	Cora	CiteSeer	PubMed	PC	Photo	CS	Physics	Cora ML	Wiki CS	ogb-arxiv
Neuromap	LIN	31.9 ± 7.9	19.5 ± 1.6	17.2 ± 12.2	0.0 ± 0.0	0.0 ± 0.0	76.7 ± 1.4	56.5 ± 1.2	24.4 ± 11.9	41.9 ± 2.3	4.7 ± 7.7
	MLP	36.6 ± 2.4	15.8 ± 1.1	23.2 ± 1.8	22.4 ± 8.6	39.8 ± 11.2	**78.1 ± 2.0**	55.9 ± 2.2	34.5 ± 1.6	43.9 ± 2.4	32.9 ± 1.3
	GCN	38.5 ± 2.0	16.2 ± 1.3	20.8 ± 1.2	**52.1 ± 2.3**	**65.0 ± 3.8**	70.4 ± 1.6	50.3 ± 1.4	36.4 ± 1.2	41.1 ± 3.2	42.3 ± 1.7
	GIN	**46.1 ± 2.0**	23.1 ± 1.2	**24.6 ± 1.7**	43.2 ± 2.4	58.2 ± 6.0	69.1 ± 2.8	52.7 ± 2.7	37.5 ± 1.9	34.9 ± 4.5	28.0 ± 2.8
	SAGE	41.6 ± 1.9	18.3 ± 1.2	20.4 ± 1.0	43.6 ± 4.8	48.1 ± 5.8	70.9 ± 2.4	51.9 ± 2.2	39.1 ± 2.1	**44.8 ± 2.2**	**43.6 ± 1.0**
DMoN	LIN	1.2 ± 1.6	3.2 ± 1.6	0.7 ± 1.0	3.4 ± 2.9	3.7 ± 4.0	25.2 ± 10.8	5.7 ± 7.3	0.4 ± 0.8	4.8 ± 3.9	4.0 ± 3.6
	MLP	36.6 ± 1.2	22.5 ± 0.9	6.7 ± 4.1	15.4 ± 11.6	43.5 ± 8.9	40.2 ± 6.5	34.7 ± 3.5	36.2 ± 1.2	31.5 ± 8.0	12.8 ± 1.7
	GCN	38.6 ± 0.9	20.4 ± 0.9	18.3 ± 0.2	43.8 ± 0.3	50.5 ± 0.3	53.1 ± 0.2	33.4 ± 0.2	37.7 ± 0.9	36.7 ± 0.4	38.4 ± 0.2
	GIN	37.2 ± 1.1	19.5 ± 0.9	17.0 ± 0.3	35.4 ± 2.3	45.6 ± 0.8	49.9 ± 0.4	29.8 ± 0.4	34.7 ± 1.5	29.7 ± 0.8	26.2 ± 3.2
	SAGE	39.1 ± 1.0	21.3 ± 1.3	17.7 ± 0.2	37.4 ± 11.9	50.8 ± 0.3	53.1 ± 0.3	33.6 ± 0.2	40.5 ± 1.0	36.4 ± 0.4	38.1 ± 0.3
NOCD	LIN	5.1 ± 1.9	2.4 ± 1.3	3.7 ± 1.7	1.2 ± 2.7	0.1 ± 0.3	40.2 ± 9.8	19.3 ± 7.0	3.3 ± 1.1	10.8 ± 3.1	4.2 ± 2.5
	MLP	**46.1 ± 1.1**	**34.8 ± 0.9**	20.5 ± 2.4	45.5 ± 2.5	60.5 ± 3.4	73.9 ± 1.7	50.2 ± 1.0	**46.5 ± 1.4**	40.9 ± 1.2	9.7 ± 2.3
	GCN	41.1 ± 0.9	20.2 ± 0.8	21.2 ± 0.6	**51.9 ± 1.9**	63.2 ± 0.9	63.3 ± 1.0	42.0 ± 0.7	39.1 ± 1.0	**44.2 ± 0.6**	**43.8 ± 0.4**
	GIN	43.4 ± 1.1	26.4 ± 1.2	21.3 ± 0.6	34.2 ± 1.9	55.3 ± 1.9	63.7 ± 0.9	48.3 ± 1.5	43.3 ± 1.2	37.6 ± 1.1	28.7 ± 1.2
	SAGE	40.7 ± 0.9	19.8 ± 0.8	19.6 ± 0.4	**51.1 ± 1.6**	**63.7 ± 1.6**	63.7 ± 0.8	42.4 ± 0.7	40.6 ± 1.1	42.4 ± 0.4	43.1 ± 0.3
DiffPool	LIN	0.4 ± 1.1	1.1 ± 1.9	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	22.2 ± 14.3	18.9 ± 15.9	0.0 ± 0.1	2.4 ± 2.1	OOM
	MLP	9.0 ± 1.2	6.7 ± 1.0	18.9 ± 1.4	1.6 ± 0.4	3.1 ± 0.9	64.5 ± 4.8	49.1 ± 4.4	9.5 ± 1.2	25.8 ± 2.6	OOM
	GCN	39.8 ± 2.0	18.2 ± 1.3	**23.7 ± 1.5**	37.0 ± 8.0	53.8 ± 7.5	73.3 ± 2.0	**62.4 ± 3.6**	33.5 ± 1.2	35.7 ± 3.8	OOM
	GIN	30.9 ± 3.6	21.3 ± 2.3	16.3 ± 3.9	14.5 ± 3.2	32.8 ± 4.6	50.3 ± 3.9	41.0 ± 3.9	31.6 ± 2.8	11.3 ± 3.1	OOM
	SAGE	38.3 ± 2.4	16.4 ± 1.4	21.9 ± 1.5	3.1 ± 0.3	16.7 ± 6.1	71.4 ± 3.4	59.2 ± 3.3	29.6 ± 2.2	38.1 ± 2.8	OOM
MinCut	LIN	26.1 ± 2.5	16.5 ± 1.3	2.2 ± 1.0	0.0 ± 0.0	0.0 ± 0.0	57.5 ± 0.6	36.8 ± 0.9	30.4 ± 1.2	16.0 ± 1.4	3.8 ± 3.5
	MLP	34.7 ± 6.6	20.5 ± 0.9	5.1 ± 2.4	0.0 ± 0.0	5.2 ± 9.9	53.9 ± 6.4	29.4 ± 9.9	39.6 ± 1.3	28.1 ± 4.2	8.0 ± 2.7
	GCN	26.2 ± 1.5	12.0 ± 0.7	11.0 ± 2.8	15.9 ± 21.2	45.2 ± 13.6	44.7 ± 3.3	32.2 ± 6.8	26.2 ± 0.7	32.1 ± 0.5	33.9 ± 4.7
	GIN	39.3 ± 3.8	18.9 ± 0.7	19.8 ± 4.3	6.8 ± 6.8	35.5 ± 19.3	32.7 ± 19.5	51.7 ± 5.4	6.9 ± 4.5	1.3 ± 6.2	10.3 ± 7.3
	SAGE	31.5 ± 1.5	16.2 ± 0.8	15.5 ± 0.3	41.6 ± 0.6	46.9 ± 0.5	47.1 ± 0.5	30.5 ± 2.4	35.9 ± 0.7	32.3 ± 0.5	33.0 ± 1.5
Ortho	LIN	7.0 ± 0.7	5.6 ± 0.7	16.5 ± 0.4	18.8 ± 9.0	24.7 ± 11.4	45.4 ± 2.0	26.7 ± 1.5	15.6 ± 1.1	29.6 ± 0.9	19.1 ± 0.5
	MLP	5.3 ± 0.5	3.8 ± 0.4	10.6 ± 0.4	19.8 ± 5.5	30.6 ± 4.0	56.3 ± 1.7	32.0 ± 1.1	5.3 ± 0.5	28.8 ± 0.9	18.5 ± 0.4
	GCN	22.9 ± 0.9	10.9 ± 0.7	13.8 ± 0.5	40.7 ± 0.9	46.0 ± 1.4	51.6 ± 1.0	32.4 ± 1.0	19.2 ± 1.1	32.1 ± 0.5	34.3 ± 0.7
	GIN	30.7 ± 1.3	17.8 ± 0.9	13.6 ± 0.5	34.0 ± 0.9	42.3 ± 0.7	44.1 ± 0.6	25.0 ± 0.7	28.7 ± 1.1	29.4 ± 0.6	24.1 ± 0.8
	SAGE	18.0 ± 0.9	8.4 ± 0.8	9.9 ± 0.4	35.4 ± 3.9	46.3 ± 1.6	52.8 ± 1.2	34.4 ± 1.1	16.3 ± 1.1	31.7 ± 0.5	24.9 ± 0.3
Infomap		35.2 ± 0.2	23.6 ± 0.1	16.0 ± 0.1	49.5 ± 0.5	57.3 ± 1.1	40.6 ± 0.2	26.7 ± 0.1	35.7 ± 0.3	37.9 ± 0.2	35.9 ± 0.1

C.3 Significance of best Neuromap results vs. best baseline

Table 3: Independent two-sample t-test between 25 samples of AMI values for Neuromap vs. the best baseline for $4\sqrt{n}$ hidden features and $s = \sqrt{n}$. The p-values indicate for which datasets the null hypothesis "the samples have the same mean" can be rejected. Blue p-values highlight cases where Neuromap performs significantly better than the best baseline; red p-values highlight cases where the best baseline performs significantly better than Neuromap.

Dataset	Best Neuromap		Best Baseline		p
Cora	GIN	46.1 ± 2.0	NOCD MLP	46.1 ± 1.1	0.92
CiteSeer	GIN	23.1 ± 1.2	NOCD MLP	34.8 ± 0.9	**$3.3e^{-35}$**
PubMed	GIN	24.6 ± 1.7	DiffPool GCN	23.7 ± 1.5	0.05
PC	GCN	52.1 ± 2.3	NOCD GCN	51.9 ± 1.9	0.69
Photo	GCN	65.0 ± 3.8	NOCD SAGE	63.7 ± 1.6	0.14
CS	MLP	78.1 ± 2.0	NOCD MLP	73.9 ± 1.7	**$4.4e^{-10}$**
Physics	LIN	56.5 ± 1.2	DiffPool GCN	62.4 ± 3.6	**$1.5e^{-08}$**
Cora ML	SAGE	39.1 ± 2.1	NOCD MLP	46.5 ± 1.4	**$5.7e^{-18}$**
Wiki CS	SAGE	44.8 ± 2.2	NOCD GCN	44.2 ± 0.6	0.23
ogb-arxiv	SAGE	43.6 ± 1.0	NOCD GCN	43.8 ± 0.4	0.38

C.4 Significance of Neuromap results vs. Infomap

Table 4: Independent two-sample t-test between 25 samples of AMI values for Neuromap with different (G)NN architectures vs. Infomap for $4\sqrt{n}$ hidden features and $s = \sqrt{n}$. The p-values indicate when the null hypothesis "the samples have the same mean" can be rejected. Blue p-values highlight cases where Neuromap performs significantly better than Infomap; red p-values highlight cases where Infomap performs significantly better than Neuromap.

Dataset	LIN	MLP	GCN	GIN	SAGE
Cora	0.05	$6.8\mathrm{e}^{-3}$	$2.4\mathrm{e}^{-8}$	$8.1\mathrm{e}^{-20}$	$9.4\mathrm{e}^{-15}$
CiteSeer	$7.5\mathrm{e}^{-12}$	$4.2\mathrm{e}^{-22}$	$2.8\mathrm{e}^{-20}$	0.06	$2.1\mathrm{e}^{-17}$
PubMed	0.65	$2.1\mathrm{e}^{-16}$	$4.2\mathrm{e}^{-16}$	$2.5\mathrm{e}^{-18}$	$2.2\mathrm{e}^{-17}$
PC	$1.8\mathrm{e}^{-48}$	$4.7\mathrm{e}^{-14}$	$6.0\mathrm{e}^{-6}$	$2.4\mathrm{e}^{-14}$	$3.9\mathrm{e}^{-6}$
Photo	$1.8\mathrm{e}^{-42}$	$6.6\mathrm{e}^{-8}$	$2.3\mathrm{e}^{-10}$	0.48	$5.2\mathrm{e}^{-8}$
CS	$7.6\mathrm{e}^{-36}$	$1.1\mathrm{e}^{-32}$	$2.9\mathrm{e}^{-32}$	$7.5\mathrm{e}^{-26}$	$5.1\mathrm{e}^{-28}$
Physics	$2.7\mathrm{e}^{-35}$	$1.2\mathrm{e}^{-28}$	$5.8\mathrm{e}^{-31}$	$3.2\mathrm{e}^{-25}$	$2.7\mathrm{e}^{-27}$
Cora ML	$9.6\mathrm{e}^{-5}$	$1.4\mathrm{e}^{-3}$	0.01	$7.0\mathrm{e}^{-5}$	$1.7\mathrm{e}^{-8}$
Wiki CS	$7.0\mathrm{e}^{-9}$	$8.3\mathrm{e}^{-12}$	$4.5\mathrm{e}^{-5}$	$3.9\mathrm{e}^{-3}$	$3.4\mathrm{e}^{-14}$
ogb-arxiv	$1.8\mathrm{e}^{-16}$	$1.3\mathrm{e}^{-11}$	$5.4\mathrm{e}^{-16}$	$4.6\mathrm{e}^{-13}$	$8.9\mathrm{e}^{-23}$

C.5 Tabulated number of detected communities on real-world networks

Here, we report the average number of detected communities for Neuromap, DMoN, NOCD, DiffPool, MinCut, Ortho, and Infomap on real-world networks for setting the number of hidden dimensions to $4\sqrt{n}$ and the maximum number of communities to $s = \sqrt{n}$.

Table 5: Average number of detected communities and their standard deviations on real-world networks. We tested each method with 2-layer MLP, GCN, GIN, and SAGE architectures, except for Infomap, which is not based on deep learning. OOM stands for "out of memory".

Method	Arch.	Cora	CiteSeer	PubMed	PC	Photo	CS	Physics	Cora ML	Wiki CS	ogb-arxiv
Neuromap	LIN	14.4 ± 3.7	38.0 ± 3.9	12.4 ± 8.1	1.0 ± 0.0	1.0 ± 0.0	50.1 ± 5.2	34.5 ± 4.5	11.4 ± 2.5	17.0 ± 1.7	2.4 ± 1.6
	MLP	44.4 ± 2.5	57.3 ± 0.9	30.8 ± 3.8	3.0 ± 0.8	4.0 ± 1.2	22.2 ± 2.4	35.8 ± 3.3	48.3 ± 1.9	12.3 ± 1.3	29.5 ± 4.4
	GCN	50.7 ± 0.9	58.0 ± 0.0	51.6 ± 4.4	21.2 ± 3.4	19.6 ± 2.1	34.6 ± 4.5	73.0 ± 8.0	54.3 ± 0.7	17.5 ± 2.4	34.4 ± 7.0
	GIN	20.8 ± 2.2	32.0 ± 2.7	12.7 ± 1.3	6.4 ± 1.3	6.8 ± 0.8	14.2 ± 2.6	14.9 ± 1.8	22.6 ± 2.7	6.8 ± 1.7	5.5 ± 1.3
	SAGE	43.9 ± 2.4	57.6 ± 0.6	34.4 ± 2.5	7.2 ± 1.9	8.1 ± 1.3	22.0 ± 2.8	41.3 ± 6.1	45.4 ± 2.8	17.1 ± 2.0	39.5 ± 4.4
DMoN	LIN	13.6 ± 13.3	25.1 ± 10.8	2.5 ± 1.7	2.4 ± 1.0	2.6 ± 1.4	10.0 ± 6.3	7.7 ± 6.7	4.5 ± 6.2	10.9 ± 5.6	7.6 ± 6.4
	MLP	52.0 ± 0.2	57.9 ± 0.3	24.8 ± 33.0	51.4 ± 36.3	81.2 ± 15.0	63.1 ± 17.1	76.3 ± 6.7	55.0 ± 0.2	92.9 ± 27.1	118.9 ± 13.2
	GCN	51.9 ± 0.3	58.0 ± 0.0	140.0 ± 0.0	117.0 ± 0.0	87.0 ± 0.0	135.0 ± 0.0	186.0 ± 0.0	54.8 ± 0.4	108.0 ± 0.0	412.0 ± 0.0
	GIN	51.6 ± 0.6	57.0 ± 1.0	140.0 ± 0.0	110.6 ± 6.1	84.5 ± 1.7	134.9 ± 0.3	185.9 ± 0.3	54.5 ± 0.6	97.2 ± 4.0	301.8 ± 77.3
	SAGE	51.8 ± 0.4	58.0 ± 0.0	140.0 ± 0.2	116.8 ± 0.7	86.8 ± 0.4	135.0 ± 0.0	186.0 ± 0.0	54.8 ± 0.4	108.0 ± 0.0	412.0 ± 0.0
NOCD	LIN	20.5 ± 10.2	17.5 ± 21.5	12.1 ± 3.1	1.7 ± 1.0	1.0 ± 0.2	34.7 ± 8.5	21.6 ± 2.4	24.8 ± 3.9	11.2 ± 0.9	12.9 ± 2.8
	MLP	29.3 ± 2.7	23.6 ± 4.1	29.8 ± 2.7	13.9 ± 3.9	11.0 ± 2.5	20.4 ± 3.7	22.8 ± 2.7	27.6 ± 1.7	28.3 ± 2.0	145.4 ± 41.7
	GCN	51.7 ± 0.6	58.0 ± 0.0	35.1 ± 2.9	72.3 ± 19.2	51.1 ± 11.0	68.4 ± 8.6	73.9 ± 7.3	54.0 ± 0.8	33.8 ± 3.9	69.0 ± 6.5
	GIN	24.2 ± 1.7	28.8 ± 2.6	35.9 ± 6.2	33.1 ± 5.8	25.1 ± 4.0	27.0 ± 4.1	29.9 ± 4.3	24.9 ± 2.6	45.7 ± 5.1	78.5 ± 15.6
	SAGE	47.8 ± 1.6	56.6 ± 1.3	60.6 ± 3.3	32.0 ± 3.0	25.1 ± 3.0	45.3 ± 3.6	55.7 ± 2.9	51.1 ± 1.7	46.6 ± 2.7	68.0 ± 6.6
DiffPool	LIN	2.6 ± 1.1	4.6 ± 3.3	1.0 ± 0.0	1.2 ± 0.5	1.0 ± 0.0	6.7 ± 6.1	8.1 ± 5.5	1.0 ± 0.2	4.4 ± 1.0	OOM
	MLP	51.8 ± 0.4	58.0 ± 0.2	43.6 ± 4.7	2.2 ± 0.5	2.2 ± 0.4	16.5 ± 2.6	39.8 ± 6.1	55.0 ± 0.0	10.0 ± 1.4	OOM
	GCN	48.2 ± 1.6	57.9 ± 0.4	15.6 ± 2.1	5.7 ± 1.1	5.7 ± 0.8	12.8 ± 2.3	19.0 ± 3.3	54.1 ± 0.9	7.7 ± 1.4	OOM
	GIN	11.5 ± 2.2	17.4 ± 3.1	7.3 ± 1.6	3.5 ± 0.6	5.2 ± 0.8	7.0 ± 1.1	6.6 ± 1.1	15.5 ± 2.5	4.7 ± 1.0	OOM
	SAGE	49.7 ± 1.3	58.0 ± 0.2	26.0 ± 3.0	3.2 ± 0.4	3.9 ± 0.7	10.7 ± 1.6	15.8 ± 2.1	54.2 ± 0.7	10.1 ± 1.2	OOM
MinCut	LIN	51.3 ± 0.9	54.8 ± 1.9	40.0 ± 6.1	1.0 ± 0.2	1.1 ± 0.3	91.0 ± 3.9	89.6 ± 6.2	55.0 ± 0.0	48.5 ± 4.9	81.6 ± 93.0
	MLP	51.5 ± 1.6	58.0 ± 0.0	25.7 ± 7.4	1.0 ± 0.0	1.2 ± 0.4	21.2 ± 41.8	118.6 ± 24.6	38.1 ± 5.8	101.4 ± 19.2	155.0 ± 49.5
	GCN	52.0 ± 0.0	58.0 ± 0.0	134.9 ± 4.1	69.5 ± 25.2	62.8 ± 20.3	133.2 ± 2.9	157.0 ± 40.2	55.0 ± 0.0	108.0 ± 0.0	97.4 ± 20.9
	GIN	21.5 ± 20.0	55.7 ± 8.5	15.0 ± 37.5	11.0 ± 4.2	10.1 ± 5.5	73.9 ± 63.8	6.1 ± 6.7	40.6 ± 6.0	4.0 ± 14.5	10.5 ± 4.2
	SAGE	52.0 ± 0.0	58.0 ± 0.0	140.0 ± 0.0	115.9 ± 2.8	86.9 ± 0.3	135.0 ± 0.0	176.5 ± 24.6	55.0 ± 0.0	108.0 ± 0.0	379.5 ± 33.7
Ortho	LIN	48.7 ± 1.3	53.4 ± 1.7	68.1 ± 5.7	9.5 ± 4.1	9.4 ± 5.1	97.4 ± 6.6	149.5 ± 5.4	34.5 ± 3.3	33.2 ± 2.6	32.8 ± 3.4
	MLP	52.0 ± 0.0	58.0 ± 0.0	126.9 ± 4.7	8.8 ± 2.3	11.2 ± 2.0	48.1 ± 9.1	101.7 ± 11.2	55.0 ± 0.0	41.0 ± 6.2	187.7 ± 18.9
	GCN	52.0 ± 0.0	58.0 ± 0.0	140.0 ± 0.0	114.9 ± 1.6	85.0 ± 3.3	115.6 ± 11.9	166.2 ± 15.9	55.0 ± 0.0	108.0 ± 0.0	253.8 ± 68.3
	GIN	52.0 ± 0.0	58.0 ± 0.0	140.0 ± 0.0	95.8 ± 4.4	78.8 ± 2.8	135.0 ± 0.0	185.8 ± 1.0	55.0 ± 0.0	107.5 ± 0.8	170.2 ± 22.4
	SAGE	52.0 ± 0.0	58.0 ± 0.0	140.0 ± 0.0	52.5 ± 30.2	52.4 ± 21.7	84.9 ± 17.5	109.5 ± 18.1	55.0 ± 0.0	105.1 ± 1.6	379.8 ± 4.6
Infomap		282.2 ± 4.1	628.9 ± 2.7	924.9 ± 9.1	455.1 ± 5.1	223.5 ± 4.4	834.4 ± 9.4	1160.8 ± 11.4	287.4 ± 3.8	747.0 ± 8.4	4840.0 ± 22.4

D Results on real-world networks with fixed number of hidden channels

We provide further results on real-world networks with a different configuration of the neural networks: we set the number of hidden channels to 512 and set the maximum number of communities to the "ground-truth" number of communities, that is $s = |Y|$.

D.1 Average Achieved AMI

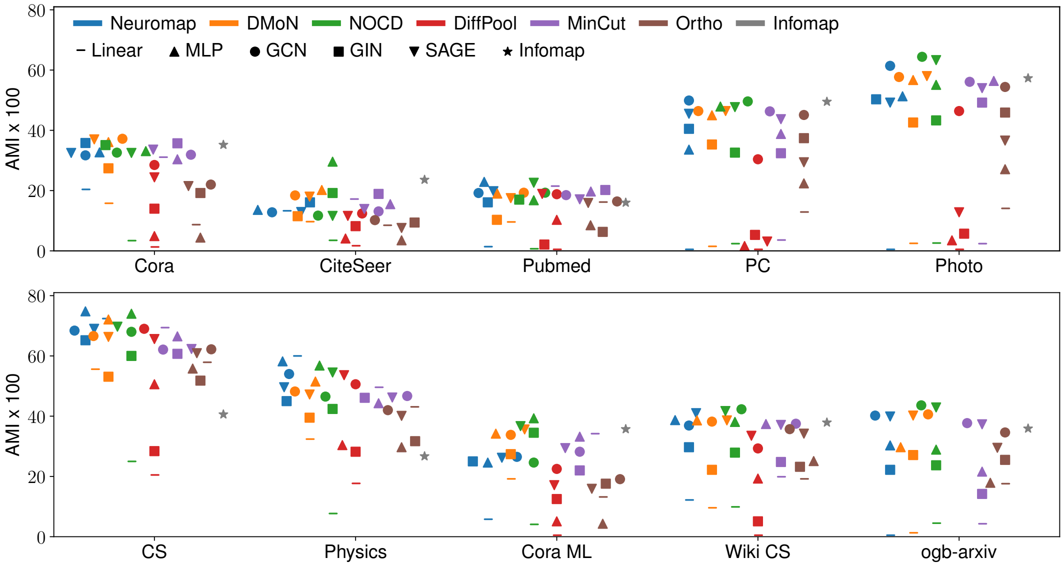


Figure 13: Average achieved AMI on real-world networks (higher is better) for $s = |Y|$, that is, the "ground-truth" number of communities. Colours indicate methods while shapes indicate neural network architectures. DiffPool ran out of memory on obg-arxiv.

D.2 Average Number of Detected Communities

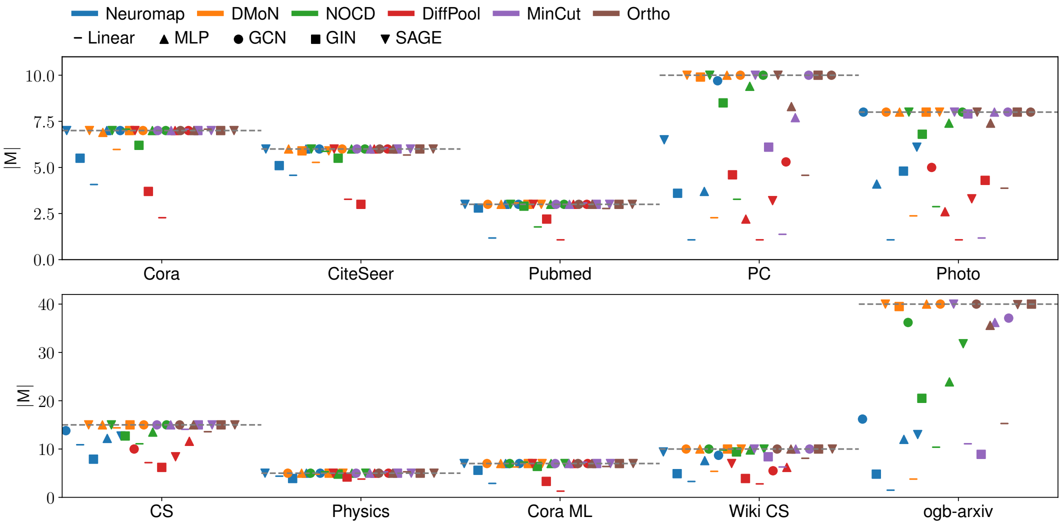


Figure 14: Average detected number of communities on real-world networks. Colours indicate methods while shapes indicate neural network architectures. The dashed horizontal lines show the correct number of communities for each dataset, which is also the maximum allowed number of communities. We omit Infomap in the plot due to the large number of detected communities.

https://doi.org/10.52202/079017-0554

D.3 Tabulated AMI results on real-world networks

Here, we report the detailed AMI performance for Neuromap, DMoN, NOCD, DiffPool, MinCut, Ortho, and Infomap on real-world networks for setting the number of hidden dimensions to 512 and the maximum number of communities to $s = |Y|$. When several of the best-performing methods achieve similar AMI, we use an independent two-sample t-test to determine whether one of them can be considered to perform better than the other. In cases where their performances do not differ significantly, we mark both as best.

Table 6: Average AMI in % (higher is better) and their standard deviations on real-world networks. For each dataset, we highlight the best scores in bold and underline the second-best score. We tested each method with 2-layer MLP, GCN, GIN, and SAGE architectures, except for Infomap, which is not based on deep learning. OOM stands for "out of memory".

Method	Arch.	Cora	CiteSeer	PubMed	PC	Photo	CS	Physics	Cora ML	Wiki CS	ogb-arxiv
Neuromap	LIN	19.9 ± 10.3	12.8 ± 6.9	0.9 ± 3.7	0.0 ± 0.0	0.0 ± 0.0	71.9 ± 3.7	$\mathbf{59.5 \pm 8.2}$	5.3 ± 9.1	11.7 ± 11.0	0.0 ± 0.0
	MLP	32.7 ± 4.6	13.6 ± 2.6	$\mathbf{22.9 \pm 4.6}$	33.6 ± 13.3	51.3 ± 11.1	$\mathbf{74.8 \pm 2.4}$	$\mathbf{58.2 \pm 6.0}$	24.6 ± 2.9	38.7 ± 4.8	30.3 ± 1.8
	GCN	31.7 ± 3.0	12.8 ± 2.9	19.2 ± 5.8	$\mathbf{49.9 \pm 3.6}$	61.4 ± 4.2	68.4 ± 2.4	54.0 ± 6.4	26.5 ± 2.8	36.9 ± 4.2	40.2 ± 1.6
	GIN	$\mathbf{35.8 \pm 4.1}$	16.1 ± 3.2	16.1 ± 5.8	40.5 ± 3.9	50.3 ± 7.0	65.2 ± 3.4	45.0 ± 6.8	25.0 ± 3.5	29.7 ± 8.1	22.2 ± 2.5
	SAGE	32.5 ± 5.0	12.9 ± 2.7	19.8 ± 4.6	45.5 ± 5.4	49.2 ± 6.1	69.0 ± 2.8	49.6 ± 6.0	26.2 ± 2.8	$\mathbf{41.0 \pm 2.5}$	39.9 ± 1.6
DMoN	LIN	15.3 ± 11.7	9.2 ± 6.9	9.1 ± 4.5	1.0 ± 0.8	2.0 ± 2.1	55.1 ± 12.9	31.9 ± 19.4	18.7 ± 12.0	9.1 ± 12.9	0.8 ± 1.1
	MLP	$\mathbf{36.1 \pm 5.2}$	20.2 ± 3.6	19.0 ± 6.5	45.0 ± 2.2	56.7 ± 3.1	72.1 ± 1.1	51.5 ± 3.9	34.2 ± 3.5	38.6 ± 3.0	29.7 ± 0.4
	GCN	$\mathbf{37.2 \pm 4.2}$	18.4 ± 2.6	19.3 ± 4.0	46.4 ± 1.9	57.7 ± 2.8	66.6 ± 1.9	48.2 ± 6.1	33.8 ± 3.5	38.2 ± 2.2	40.6 ± 0.4
	GIN	27.4 ± 3.3	11.5 ± 3.1	10.3 ± 7.1	35.3 ± 2.3	42.6 ± 4.9	53.1 ± 3.7	39.5 ± 6.4	27.4 ± 2.4	22.2 ± 3.5	27.1 ± 1.6
	SAGE	$\mathbf{37.0 \pm 4.0}$	18.0 ± 3.4	17.5 ± 3.7	46.4 ± 1.6	58.0 ± 3.2	66.3 ± 1.6	47.2 ± 4.8	$\underline{35.6 \pm 3.8}$	38.6 ± 2.1	40.2 ± 0.4
NOCD	LIN	2.9 ± 1.3	3.0 ± 1.6	0.2 ± 0.5	1.9 ± 2.5	2.1 ± 3.0	24.5 ± 7.7	7.2 ± 4.5	3.6 ± 1.2	9.4 ± 2.5	4.0 ± 1.4
	MLP	$\mathbf{33.1 \pm 14.7}$	$\mathbf{29.6 \pm 10.8}$	16.8 ± 8.1	47.9 ± 2.1	55.1 ± 15.4	$\mathbf{74.0 \pm 1.4}$	$\mathbf{56.8 \pm 3.9}$	$\mathbf{39.3 \pm 5.3}$	38.1 ± 2.5	28.9 ± 0.5
	GCN	32.6 ± 4.6	11.7 ± 1.7	$\mathbf{19.3 \pm 7.3}$	$\mathbf{49.6 \pm 2.2}$	$\mathbf{64.4 \pm 4.1}$	68.0 ± 2.3	46.5 ± 7.3	24.6 ± 11.2	$\mathbf{42.3 \pm 2.2}$	$\mathbf{43.6 \pm 0.5}$
	GIN	$\mathbf{35.1 \pm 4.8}$	19.2 ± 3.4	17.0 ± 6.3	32.6 ± 3.7	43.3 ± 4.1	60.0 ± 3.8	42.4 ± 4.9	34.5 ± 3.9	27.9 ± 3.2	23.7 ± 1.2
	SAGE	32.5 ± 3.7	11.6 ± 2.8	$\mathbf{22.6 \pm 4.9}$	47.7 ± 2.4	$\mathbf{63.3 \pm 3.5}$	69.7 ± 1.9	54.5 ± 4.7	$\underline{36.7 \pm 2.8}$	$\mathbf{41.7 \pm 2.3}$	$\underline{42.9 \pm 0.4}$
DiffPool	LIN	0.8 ± 1.9	1.2 ± 0.5	0.0 ± 0.0	0.0 ± 0.0	0.0 ± 0.0	20.0 ± 14.2	17.2 ± 18.2	0.0 ± 0.0	0.0 ± 0.0	OOM
	MLP	4.9 ± 1.3	4.1 ± 1.4	10.3 ± 3.6	1.6 ± 0.4	3.5 ± 1.4	50.6 ± 6.7	30.4 ± 7.9	5.1 ± 1.0	19.3 ± 3.4	OOM
	GCN	28.5 ± 4.1	12.4 ± 2.5	18.8 ± 5.5	30.4 ± 9.1	46.4 ± 6.5	69.0 ± 3.9	50.6 ± 6.7	22.5 ± 3.1	29.3 ± 4.8	OOM
	GIN	14.0 ± 4.8	8.2 ± 3.1	2.1 ± 1.7	5.3 ± 5.6	5.7 ± 6.8	28.4 ± 9.1	28.2 ± 12.2	12.5 ± 3.8	5.1 ± 3.0	OOM
	SAGE	24.4 ± 3.0	11.6 ± 2.5	18.9 ± 5.0	3.1 ± 0.6	12.8 ± 6.1	65.6 ± 4.5	53.6 ± 9.0	17.1 ± 2.9	33.5 ± 3.2	OOM
MinCut	LIN	30.6 ± 4.3	16.7 ± 2.6	$\mathbf{21.0} \pm 1.9$	3.1 ± 8.5	1.9 ± 6.4	68.9 ± 3.0	49.1 ± 3.4	33.7 ± 3.4	19.4 ± 10.4	3.8 ± 2.0
	MLP	30.4 ± 4.5	15.5 ± 3.5	$\mathbf{19.7 \pm 6.3}$	38.8 ± 13.3	56.4 ± 2.7	66.5 ± 1.2	44.3 ± 4.8	33.2 ± 4.7	37.4 ± 2.3	21.6 ± 8.1
	GCN	31.9 ± 3.7	13.1 ± 2.8	18.5 ± 4.2	46.3 ± 2.0	56.1 ± 2.9	62.1 ± 1.4	46.7 ± 5.3	28.2 ± 3.2	37.5 ± 1.9	37.7 ± 1.4
	GIN	$\mathbf{35.7 \pm 2.9}$	18.9 ± 3.0	20.2 ± 4.1	32.4 ± 7.8	49.2 ± 4.9	60.7 ± 1.4	46.1 ± 4.3	22.0 ± 2.4	24.8 ± 11.9	14.2 ± 4.6
	SAGE	33.6 ± 2.9	13.9 ± 3.5	17.1 ± 3.7	43.7 ± 2.5	54.0 ± 2.8	62.3 ± 1.2	46.2 ± 6.8	29.4 ± 3.7	37.1 ± 1.9	37.3 ± 0.5
Ortho	LIN	8.2 ± 1.2	8.0 ± 2.3	15.7 ± 4.6	12.4 ± 6.5	13.6 ± 8.6	57.4 ± 2.6	42.6 ± 5.1	12.7 ± 3.7	18.7 ± 5.0	17.1 ± 1.1
	MLP	4.4 ± 1.0	3.5 ± 1.0	8.5 ± 3.6	22.4 ± 5.7	27.1 ± 2.9	55.8 ± 2.7	29.7 ± 7.4	4.3 ± 1.1	25.1 ± 1.8	17.9 ± 0.5
	GCN	22.0 ± 3.0	10.2 ± 2.1	16.4 ± 4.8	45.1 ± 1.9	54.4 ± 3.6	62.2 ± 2.0	42.0 ± 5.9	19.1 ± 3.2	35.7 ± 1.9	34.6 ± 0.8
	GIN	19.2 ± 3.6	9.4 ± 2.9	6.3 ± 3.0	37.4 ± 3.8	45.9 ± 3.0	51.8 ± 2.7	31.7 ± 5.4	17.6 ± 2.6	23.2 ± 2.6	25.5 ± 1.2
	SAGE	21.5 ± 4.1	7.6 ± 2.5	15.9 ± 3.8	29.4 ± 2.3	36.6 ± 3.2	60.9 ± 2.1	40.1 ± 6.3	15.9 ± 3.2	34.2 ± 2.2	29.5 ± 0.6
Infomap		35.2 ± 0.2	$\underline{23.6 \pm 0.1}$	16.0 ± 0.1	$\mathbf{49.5 \pm 0.5}$	57.3 ± 1.1	40.6 ± 0.2	26.7 ± 0.1	$\underline{35.7 \pm 0.3}$	37.9 ± 0.2	35.9 ± 0.1

D.4 Significance of best Neuromap results vs. best baseline

Table 7: Independent two-sample t-test between 25 samples of AMI values for Neuromap vs. the best baseline for 512 hidden features and $s = |Y|$. The p-values indicate for which datasets the null hypothesis "the samples have the same mean" can be rejected. Red p-values highlight cases where the best baseline performs significantly better than Neuromap.

Dataset	Best Neuromap		Best Baseline		p
Cora	GIN	35.8 ± 4.1	DMoN GCN	37.2 ± 4.2	0.25
CiteSeer	GIN	16.1 ± 3.2	NOCD MLP	29.6 ± 10.8	$\mathbf{2.4e^{-06}}$
PubMed	MLP	22.9 ± 4.6	NOCD SAGE	22.6 ± 4.9	0.83
PC	GCN	49.9 ± 3.6	NOCD GCN	49.6 ± 2.2	0.73
Photo	GCN	61.4 ± 4.2	NOCD GCN	64.4 ± 4.1	0.02
CS	MLP	74.8 ± 2.4	NOCD MLP	74.0 ± 1.4	0.15
Physics	LIN	59.5 ± 8.2	NOCD MLP	56.8 ± 3.9	0.17
Cora ML	GCN	26.5 ± 2.8	NOCD MLP	39.3 ± 5.3	$\mathbf{1.6e^{-12}}$
Wiki CS	SAGE	41.0 ± 2.5	NOCD GCN	42.3 ± 2.2	0.05
ogb-arxiv	GCN	40.2 ± 1.6	NOCD GCN	43.6 ± 0.5	$\mathbf{4.8e^{-11}}$

D.5 Significance of Neuromap results vs. Infomap

Table 8: Independent two-sample t-test between 25 samples of AMI values for Neuromap with different (G)NN architectures vs. Infomap for 512 hidden features and $s = |Y|$. The p-values indicate when the null hypothesis "the samples have the same mean" can be rejected. Blue p-values highlight cases where Neuromap performs significantly better than Infomap; red p-values highlight cases where Infomap performs significantly better than Neuromap.

Dataset	LIN	MLP	GCN	GIN	SAGE
Cora	$\mathbf{1.6e^{-7}}$	0.01	$\mathbf{9.2e^{-6}}$	0.47	0.02
CiteSeer	$\mathbf{6.4e^{-8}}$	$\mathbf{1.0e^{-15}}$	$\mathbf{1.1e^{-15}}$	$\mathbf{2.7e^{-11}}$	$\mathbf{5.1e^{-16}}$
PubMed	$\mathbf{1.2e^{-16}}$	$\mathbf{1.5e^{-7}}$	0.01	0.95	$\mathbf{4.9e^{-4}}$
PC	$\mathbf{1.8e^{-48}}$	$\mathbf{5.1e^{-6}}$	0.55	$\mathbf{4.4e^{-11}}$	$\mathbf{1.6e^{-3}}$
Photo	$\mathbf{1.8e^{-42}}$	0.02	$\mathbf{6.3e^{-5}}$	$\mathbf{5.7e^{-5}}$	$\mathbf{1.1e^{-6}}$
CS	$\mathbf{6.2e^{-24}}$	$\mathbf{1.8e^{-29}}$	$\mathbf{3.0e^{-27}}$	$\mathbf{2.5e^{-22}}$	$\mathbf{8.8e^{-26}}$
Physics	$\mathbf{3.1e^{-16}}$	$\mathbf{6.5e^{-19}}$	$\mathbf{7.3e^{-17}}$	$\mathbf{1.6e^{-12}}$	$\mathbf{7.2e^{-16}}$
Cora ML	$\mathbf{1.7e^{-14}}$	$\mathbf{5.3e^{-16}}$	$\mathbf{2.5e^{-14}}$	$\mathbf{9.9e^{-14}}$	$\mathbf{6.5e^{-15}}$
Wiki CS	$\mathbf{2.4e^{-11}}$	0.42	0.29	$\mathbf{5.3e^{-5}}$	$\mathbf{3.4e^{-6}}$
ogb-arxiv	$\mathbf{3.7e^{-62}}$	$\mathbf{4.5e^{-14}}$	$\mathbf{2.0e^{-12}}$	$\mathbf{1.3e^{-19}}$	$\mathbf{5.4e^{-12}}$

D.6 Tabulated number of detected communities on real-world networks

Here, we report the average number of detected communities for Neuromap, DMoN, NOCD, DiffPool, MinCut, Ortho, and Infomap on real-world networks for setting the number of hidden dimensions to 512 and the maximum number of communities to $s = |Y|$.

Table 9: Average number of detected communities and their standard deviations on real-world networks. We tested each method with 2-layer MLP, GCN, GIN, and SAGE architectures, except for Infomap, which is not based on deep learning. OOM stands for "out of memory".

Method	Arch.	Cora	CiteSeer	PubMed	PC	Photo	CS	Physics	Cora ML	Wiki CS	ogb-arxiv
Neuromap	LIN	4.0 ± 1.2	4.5 ± 0.9	1.1 ± 0.3	1.0 ± 0.2	1.0 ± 0.0	10.6 ± 1.4	4.1 ± 0.7	2.6 ± 0.8	3.0 ± 0.9	1.2 ± 0.4
	MLP	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	3.7 ± 0.9	4.1 ± 1.1	12.2 ± 1.2	5.0 ± 0.0	7.0 ± 0.0	7.6 ± 1.1	12.0 ± 2.4
	GCN	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	9.7 ± 0.5	8.0 ± 0.2	13.8 ± 1.0	5.0 ± 0.0	7.0 ± 0.0	8.7 ± 0.9	16.2 ± 2.3
	GIN	5.5 ± 0.8	5.1 ± 0.6	2.8 ± 0.4	3.6 ± 0.5	4.8 ± 0.9	7.9 ± 0.8	3.9 ± 0.7	5.6 ± 0.8	4.9 ± 1.2	4.8 ± 1.5
	SAGE	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	6.5 ± 0.9	6.1 ± 0.5	12.7 ± 1.2	5.0 ± 0.0	7.0 ± 0.0	9.4 ± 0.6	13.0 ± 1.1
DMoN	LIN	5.9 ± 1.8	5.2 ± 1.5	3.0 ± 0.0	2.2 ± 0.9	2.3 ± 1.0	14.1 ± 2.5	4.5 ± 0.9	6.0 ± 1.8	5.1 ± 3.1	3.5 ± 3.2
	MLP	6.9 ± 0.3	6.0 ± 0.0	3.0 ± 0.0	10.0 ± 0.0	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.2	10.0 ± 0.0	40.0 ± 0.0
	GCN	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	10.0 ± 0.0	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.2	7.0 ± 0.2	10.0 ± 0.0	40.0 ± 0.0
	GIN	7.0 ± 0.0	5.9 ± 0.3	3.0 ± 0.0	9.9 ± 0.3	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.2	39.5 ± 1.0
	SAGE	7.0 ± 0.2	5.9 ± 0.3	3.0 ± 0.0	10.0 ± 0.0	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	40.0 ± 0.0
NOCD	LIN	7.0 ± 0.0	5.8 ± 0.4	1.7 ± 0.5	3.2 ± 1.1	2.8 ± 1.2	10.8 ± 2.5	5.0 ± 0.0	7.0 ± 0.0	9.5 ± 0.7	10.1 ± 1.3
	MLP	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	9.4 ± 0.9	7.4 ± 0.6	13.5 ± 1.1	5.0 ± 0.0	7.0 ± 0.2	9.8 ± 0.6	23.9 ± 3.6
	GCN	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	10.0 ± 0.0	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	36.2 ± 1.4
	GIN	6.2 ± 0.6	5.5 ± 0.6	2.9 ± 0.3	8.5 ± 0.6	6.8 ± 1.0	12.7 ± 1.4	4.8 ± 0.4	6.4 ± 0.6	9.4 ± 0.6	20.5 ± 3.2
	SAGE	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	10.0 ± 0.2	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	31.8 ± 1.8
DiffPool	LIN	2.2 ± 0.7	3.2 ± 0.6	1.0 ± 0.0	1.0 ± 0.0	1.0 ± 0.0	6.9 ± 2.5	3.5 ± 1.0	1.0 ± 0.0	2.5 ± 0.6	OOM
	MLP	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	2.2 ± 0.4	2.6 ± 0.7	11.6 ± 1.9	5.0 ± 0.0	7.0 ± 0.0	6.2 ± 0.9	OOM
	GCN	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	5.3 ± 1.4	5.0 ± 0.7	10.0 ± 1.2	5.0 ± 0.0	7.0 ± 0.0	5.5 ± 1.0	OOM
	GIN	3.7 ± 0.6	3.0 ± 0.7	2.2 ± 0.4	4.6 ± 1.2	4.3 ± 1.1	6.2 ± 3.1	4.2 ± 0.9	3.3 ± 0.9	3.9 ± 1.3	OOM
	SAGE	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	3.2 ± 0.4	3.3 ± 0.5	8.4 ± 1.3	5.0 ± 0.0	7.0 ± 0.0	7.0 ± 0.8	OOM
MinCut	LIN	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	1.3 ± 0.7	1.1 ± 0.3	13.8 ± 1.1	5.0 ± 0.0	7.0 ± 0.0	6.0 ± 2.4	10.8 ± 9.1
	MLP	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	7.7 ± 3.4	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	36.2 ± 4.6
	GCN	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	10.0 ± 0.0	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	37.1 ± 5.4
	GIN	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	6.1 ± 1.8	7.9 ± 0.3	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	8.4 ± 3.4	8.9 ± 3.8
	SAGE	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	10.0 ± 0.0	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	40.0 ± 0.0
Ortho	LIN	7.0 ± 0.2	5.6 ± 0.5	2.7 ± 0.4	4.5 ± 1.5	3.8 ± 1.4	13.3 ± 0.6	5.0 ± 0.0	6.1 ± 1.4	7.8 ± 1.7	15.0 ± 1.7
	MLP	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	8.3 ± 1.5	7.4 ± 0.7	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	35.6 ± 1.8
	GCN	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	10.0 ± 0.0	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	40.0 ± 0.0
	GIN	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	10.0 ± 0.0	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	40.0 ± 0.0
	SAGE	7.0 ± 0.0	6.0 ± 0.0	3.0 ± 0.0	10.0 ± 0.0	8.0 ± 0.0	15.0 ± 0.0	5.0 ± 0.0	7.0 ± 0.0	10.0 ± 0.0	39.9 ± 0.3
Infomap		282.2 ± 4.1	628.9 ± 2.7	924.9 ± 9.1	455.1 ± 5.1	223.5 ± 4.4	834.4 ± 9.4	1160.8 ± 11.4	287.4 ± 3.8	747.0 ± 8.4	4840.0 ± 22.4

https://doi.org/10.52202/079017-0554

E Performance impact of hidden layers and max. number of communities

Here, we report the performance difference in terms of achieved average AMI score between using (i) $4\sqrt{n}$ hidden layers and a maximum of $s = \sqrt{n}$ communities, and (ii) 512 hidden layers and a maximum of $s = |Y|$ communities, that is, the "ground-truth" number of communities. The tabulated values show much better or worse the methods perform when using setup (ii) as compared to setup (i). That is, a positive value, shown in green, means that setup (ii) gives a better result while a negative value, shown in red, means that setup (i) gives a better result.

We find that, for Neuromap, NOCD, and DiffPool, setup (i) generally works better. In contrast, DMoN, MinCut, and Ortho often perform better with setup (ii), indicating that they require knowing the correct number of communities to perform well.

Table 10: Performance difference in terms of achieved average AMI score between (i) $4\sqrt{n}$ hidden layers and a maximum of $s = \sqrt{n}$ communities, and (ii) 512 hidden layers and a maximum of $s = |Y|$ communities, that is, the "ground-truth" number of communities. OOM stands for "out of memory".

Method	Arch.	Cora	CiteSeer	PubMed	PC	Photo	CS	Physics	Cora ML	Wiki CS	ogb-arxiv
Neuromap	LIN	−12.00	−6.70	−16.30	0.00	0.00	−4.80	+3.00	−19.10	−30.20	−4.70
	MLP	−3.90	−2.20	−0.30	+11.20	+11.50	−3.30	+2.30	−9.90	−5.20	−2.60
	GCN	−6.80	−3.40	−1.60	−2.20	−3.60	−2.00	+3.70	−9.90	−4.20	−2.10
	GIN	−10.30	−7.00	−8.50	−2.70	−7.90	−3.90	−7.70	−12.50	−5.20	−5.80
	SAGE	−9.10	−5.40	−0.60	+1.90	+1.10	−1.90	−2.30	−12.90	−3.80	−3.70
DMoN	LIN	+14.10	+6.00	+8.40	−2.40	−1.70	+29.90	+26.20	+18.30	+4.30	−3.20
	MLP	−0.50	−2.30	+12.30	+29.60	+13.20	+31.90	+16.80	−2.00	+7.10	+16.90
	GCN	−1.40	−2.00	+1.00	+2.60	+7.20	+13.50	+14.80	−3.90	+1.50	+2.20
	GIN	−9.80	−8.00	−6.70	−0.10	−3.00	+3.20	+9.70	−7.30	−7.50	+0.90
	SAGE	−2.10	−3.30	−0.20	+9.00	+7.20	+13.20	+13.60	−4.90	+2.20	+2.10
NOCD	LIN	−2.20	+0.60	−3.50	+0.70	+2.00	−15.70	−12.10	+0.30	−1.40	−0.20
	MLP	−13.00	−5.20	−3.70	+2.40	−5.40	+0.10	+6.60	−7.20	−2.80	+19.20
	GCN	−8.50	−8.50	−1.90	−2.30	+1.20	+4.70	+4.50	−14.50	−1.90	−0.20
	GIN	−8.30	−7.20	−4.30	−1.60	−12.00	−3.70	−5.90	−8.80	−9.70	−5.00
	SAGE	−8.20	−8.20	+3.00	−3.40	−0.40	+6.00	+12.10	−3.90	−0.70	−0.20
DiffPool	LIN	+0.40	+0.10	0.00	0.00	0.00	−2.20	−1.70	0.00	−2.40	OOM
	MLP	−4.10	−2.60	−8.60	0.00	+0.40	−13.90	−18.70	−4.40	−6.50	OOM
	GCN	−11.30	−5.80	−4.90	−6.60	−7.40	−4.30	−11.80	−11.00	−6.40	OOM
	GIN	−16.90	−13.10	−14.20	−9.20	−27.10	−21.90	−12.80	−19.10	−6.20	OOM
	SAGE	−13.90	−4.80	−3.00	0.00	−3.90	−5.80	−5.60	−12.50	−4.60	OOM
MinCut	LIN	+4.50	+0.20	+18.80	+3.10	+1.90	+11.40	+12.30	+3.30	+3.40	0.00
	MLP	−4.30	−5.00	+14.60	+38.80	+51.20	+12.60	+14.90	−6.40	+9.30	+13.60
	GCN	+5.70	+1.10	+7.50	+30.40	+10.90	+17.40	+14.50	+2.00	+5.40	+3.80
	GIN	−3.60	0.00	+0.40	+25.60	+13.70	+28.00	−5.60	+15.10	+23.50	+3.90
	SAGE	+2.10	−2.30	+1.60	+2.10	+7.10	+15.20	+15.70	−6.50	+4.80	+4.30
Ortho	LIN	+1.20	+2.40	−0.80	−6.40	−11.10	+12.00	+15.90	−2.90	−10.90	−2.00
	MLP	−0.90	−0.30	−2.10	+2.60	−3.50	−0.50	−2.30	−1.00	−3.70	−0.60
	GCN	−0.90	−0.70	+2.60	+4.40	+8.40	+10.60	+9.60	−0.10	+3.60	+0.30
	GIN	−11.50	−8.40	−7.30	+3.40	+3.60	+7.70	+6.70	−11.10	−6.20	+1.40
	SAGE	+3.50	−0.80	+6.00	−6.00	−9.70	+8.10	+5.70	−0.40	+2.50	+4.60
Infomap		0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

https://doi.org/10.52202/079017-0554

F Further results on a synthetic network with overlapping communities

Here we show the detected communities on a synthetic network with overlapping communities with a dense linear layer, a 2-layer MLP, a 2-layer GIN, and a 2-layer SAGE architecture. We use $|V|$ hidden channels and set the maximum number of communities to $s \in \{2, 3\}$.

F.1 Linear-layer-based results

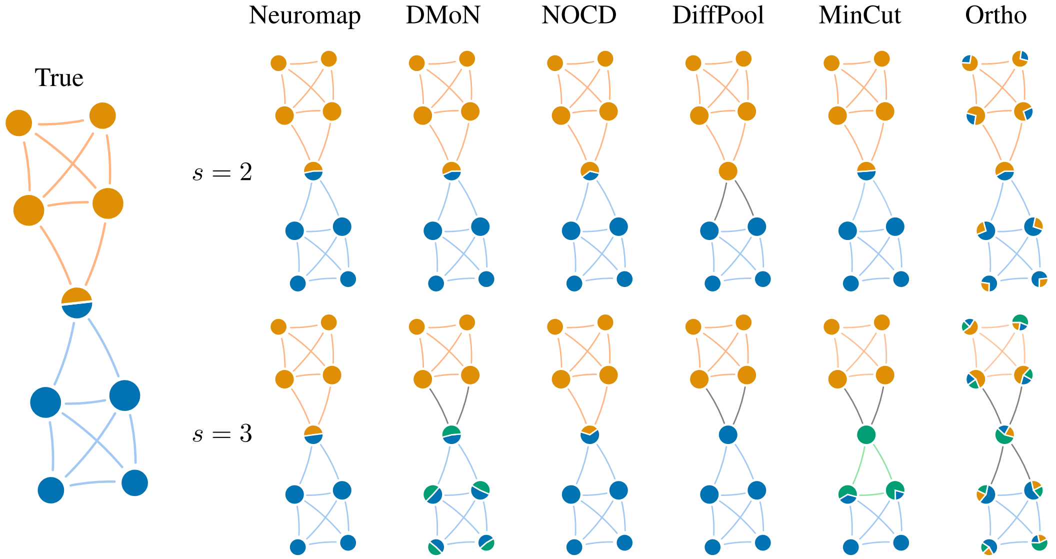


Figure 15: Linear-layer-based results on a synthetic network with overlapping communities where the leftmost network shows the true community structure. Nodes are drawn as pie charts to visualise their community assignments. The top and bottom rows show results for a maximum of $s = 2$ and $s = 3$ communities, respectively.

F.2 MLP-based results

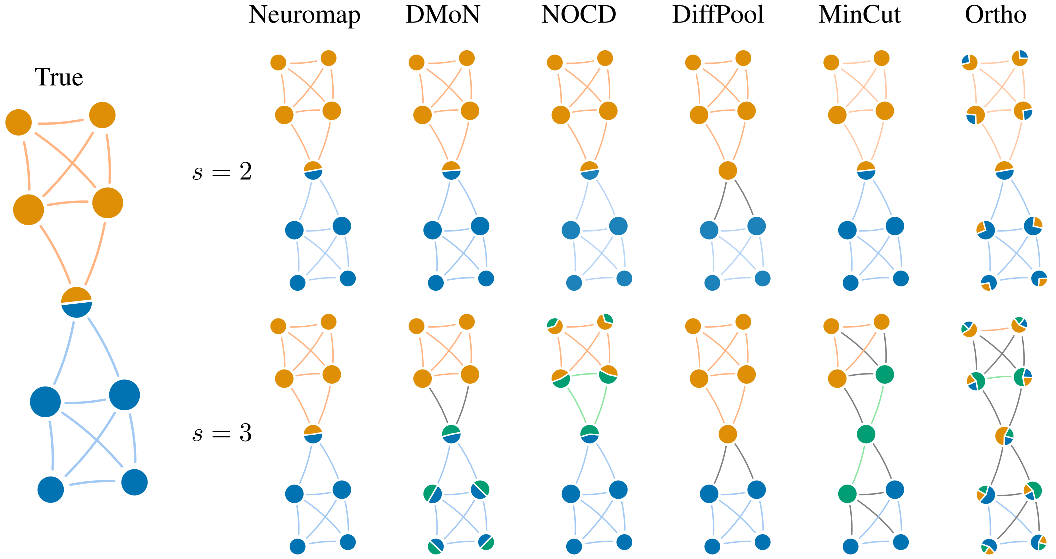


Figure 16: MLP-based results on a synthetic network with overlapping communities where the leftmost network shows the true community structure. Nodes are drawn as pie charts to visualise their community assignments. The top and bottom rows show results for a maximum of $s = 2$ and $s = 3$ communities, respectively.

https://doi.org/10.52202/079017-0554

F.3 GIN-based results

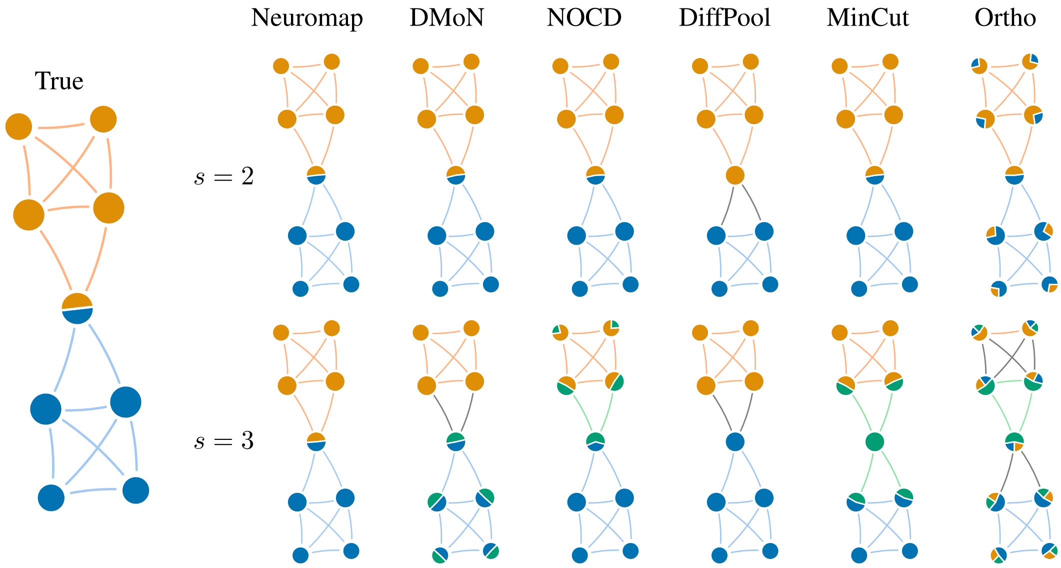


Figure 17: GIN-based results on a synthetic network with overlapping communities where the leftmost network shows the true community structure. Nodes are drawn as pie charts to visualise their community assignments. The top and bottom rows show results for a maximum of $s = 2$ and $s = 3$ communities, respectively.

F.4 SAGE-based results

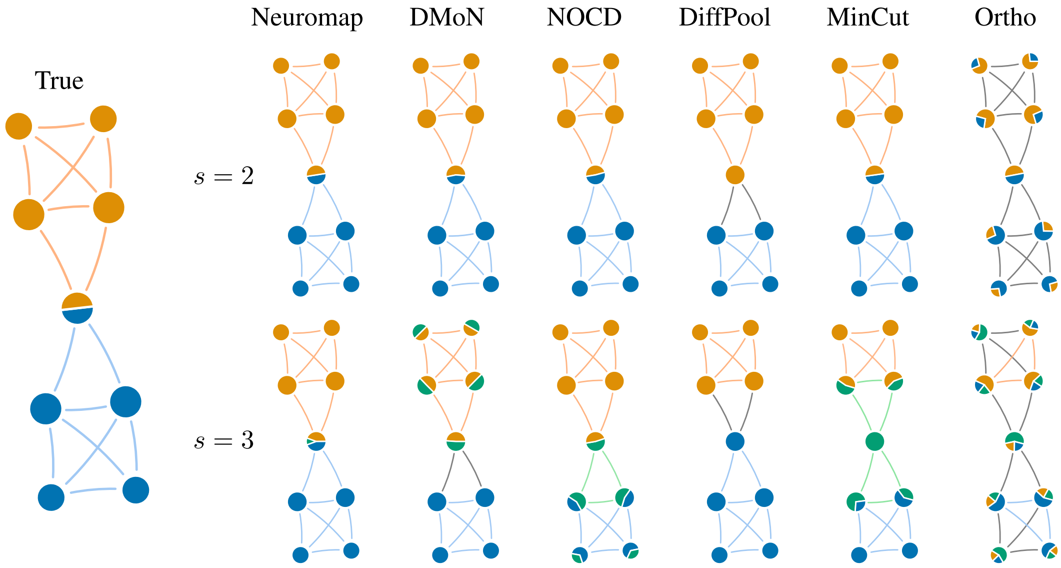


Figure 18: SAGE-based results on a synthetic network with overlapping communities where the leftmost network shows the true community structure. Nodes are drawn as pie charts to visualise their community assignments. The top and bottom rows show results for a maximum of $s = 2$ and $s = 3$ communities, respectively.

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: Abstract and introduction introduce the research gap and summarise the developed method and experimentally achieved results on a high level.

2. **Limitations**

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

   Answer: [Yes]

   Justification: We discuss the limitation of our work in section 4.

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: [NA]

   Justification: We focus on adapting the well-established map equation framework for community detection with (graph) neural networks and show its utility empirically, however, our work does not include theoretical results or proofs.

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: Our main contribution is adapting the map equation as a differentiable loss function for optimisation with (graph) neural networks. We present the mathematical formulation of our adaptation in section 4. In section 5, we describe our experimental setup, including hyperparameter settings for our experiments.

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 made our code available through a public GitHub repository and the synthetic networks through a Zenodo repository. The used real-world networks are publicly available.

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, etc.) necessary to understand the results?

   Answer: [Yes]

   Justification: We detail our experimental settings, including hyperparameters, in section 5.

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: We report the average achieved performance with standard deviations.

8. **Experiments Compute Resources**

https://doi.org/10.52202/079017-0554

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 describe the utilised computational hardware in section 5.

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 the code of ethics and find that we conform with it.

10. **Broader Impacts**

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

    Answer: [No]

    Justification: We consider the task of detecting communities in complex networks. While any algorithm can potentially be abused for malicious purposes, we believe that, in our case, this would be a rather contrived scenario.

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., pretrained language models, image generators, or scraped datasets)?

    Answer: [No]

    Justification: We find it not applicable to provide safeguards in the context of detecting communities in complex networks to understand their organisational structure.

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 the original works of the models we use as baselines as well as the sources of the datasets we use.

13. **New Assets**

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

    Answer: [NA]

    Justification: We do not introduce new assets.

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: [NA]

    Justification: Our work involves neither crowdsourcing nor research with human subjects.

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: [NA]

    Justification: Our work involves neither crowdsourcing nor research with human subjects.

https://doi.org/10.52202/079017-0554