A Framework for Comprehensive Evaluations of Graph Neural Network based Community Detection Using Unsupervised Node Clustering

Anonymous authors
Paper under double-blind review

Abstract

Graph Neural Networks (GNNs) have shown promising performance across a number of tasks in recent years. Unsupervised community detection using GNNs involves the clustering of nodes of a graph given both the features of nodes as well as the structure of the graph, and has many applications to real world tasks from social networks to genomics. Unfortunately, there has been relatively little research using GNNs for commOunity detection, and even less that evaluates the systems rigorously and fairly. A comprehensive evaluation of the performance of GNNs requires an suitable environment within which they are evaluated. This is exacerbated by the fact that community detection is primarily an unsupervised task, and that (graph) neural networks are used which contain many hyperparameters, discovered by inconsistent procedures. We argue that there is currently a gap in the literature that establishes a sufficient benchmarking environment for the consistent evaluation of GNN based community detection, thereby impeding progress in this nascent field. In this work we propose and evaluate an environment for the consistent evaluation of neural community detection. With this we show the strong dependence of the performance to the experimental settings , thereby motivating the use of this framework to facilitate research into GNN based community detection.

1 Introduction

Graphs provide an intuitive way of representing the structural connectivity of data. Graph Neural Networks (GNNs) have become popular as a neural network based approach for processing graph-structured data (Scarselli et al., 2008; Kipf & Welling, 2016a). Many real-world systems can be represented as graphs which contain clusters of similar nodes, according to some definition of similarity, Schaeffer (2007) and community detection is the task of finding these clusters. There are many definitions of clusters within graphs Edachery et al. (1999), but a popular definition is that these clusters are groups of nodes with a high in-group edge density and relatively low out-group density Tsitsulin et al. (2020). Analysing the structure to find clusters, or communities, of nodes enables us to extract useful information for solving real world problems such as fake new detection (Monti et al., 2019), genomic feature discovery (Cabreros et al., 2016), communities of similar people in social networks Yang et al. (2013) and communities of similar research papers or researchers in citation networks Chen & Redner (2010).

Traditional community detection methods typically partition the adjacency matrix by splitting the graph to form communities of similar nodes in the graphical space. In contrast, community detection performed by GNNs, via the process of node clustering, encodes the duality of graphical information alongside feature space into a single vector space representation in order to cluster the nodes into communities. However, due to a lack consistent and rigorous way of measuring performance in (unsupervised) node clustering in the literature currently, the best choice of algorithm, or process to discover it, is often unclear. There are a number of challenges with community detection, in particular given the unsupervised nature of the task, as all algorithms typically rely on hyperparameters to be set, typically tailored to a specific dataset. Further, the process of discovering these often involves testing various hyperparameter configurations on the dataset until sufficient performance is achieved. From a machine learning perspective, in the community detection setting, this may be troubling as by carrying out this process we may be overfitting to the dataset. For a practitioner with a new dataset, they are left with little choice but to repeat this process for all algorithms, across many hyperparameter values, until they are content with their result, which itself can be measured by various different metrics.

This paper proposes that, while GNN based community detection is still in it's infancy but has shown great potential, we must establish a fair and comprehensive environment for the evaluation and comparison of methods. A rigorous study of GNNs for community detection requires a consistent evaluation environment. Such a framework for evaluation should include multiple metrics of performance and a consistent training procedure, testing over datasets that represent a wide range of potential topologies and applications, as well as a well-defined hyperparameter search procedure. Previous research involving training unsupervised GNNs compare works in inconsistent and often incomplete (in terms of details) manners, making it difficult to accurately capture and compare performance.

The paper is structured as follows. In Section 2, it is demonstrated that the latest surveys of the landscape of community detection fail to capture relevant experimental investigation. They acknowledge the issues present in research, such as different works reporting on different datasets using different performance metrics do not benchmark the methods themselves. The GNNs we investigate are summarised in Section 3. Often, the evaluation of these is inconsistent due to the lack of detail provided on the hyperparameter optimisation (HPO) strategy undertaken originally, if at all. Therefore, we investigate the relevant contribution of the HPO to GNN based community detection performance. There are a variety of complex algorithms that can be used for community detection on a variety of datasets. Section 4 details, and executes, the framework to evaluate the performance of each algorithm across a suite of datasets. In Section 5 we discuss the results.

1.1 Contributions

In this paper, we make the following key contributions:

- We are the first propose a framework for the rigorous evaluation of GNN based community detection; we aim to set a standard for future research in this area.
- We open-source the framework to facilitate and encourage the use of our framework by those proposing and evaluating new methods[1].
- Through rigorous evaluation we show experimental conditions significantly impact performance of these methods, thereby demonstrating the necessity for this framework.

2 Literature Review

We will provide an overview of the literature related to community detection, highlighting that while GNNs have shown promise, there is often missing context and experimental details. We also review other evaluation frameworks proposed for GNNs, highlighting the gap that our proposed framework addresses.

2.1 Community Detection

Liu et al. (2020) argues for the use of graph deep learning for the task of community detection and details the methodology behind current approaches. Su et al. (2022) provides a taxonomy of existing methods, comparing representative works in the GNN space, as well as deep non-negative matrix factorisation and sparse filtering-based methods, but do not evaluate the methods. Jin et al. (2021) does contrast classical approaches to community detection and GNN based methods, however are focused on providing clarity to the theoretical underpinning of community detection methods. Chunaev (2020) provides an overview of community detection methods but omit any mention of GNN methods. Tu et al. (2018) provides an overview of network representation learning for community

[1]we will make the URL available upon acceptance

detection and proposes a novel method to incorporate node connectivity and features, but does not compare GNN methods in the study. Ezugwu et al. (2022) is survey of clustering algorithms providing an in-depth discussion of all applications of clustering algorithms, but neglects an investigation into methods that perform clustering on graphs.

2.2 Evaluation Frameworks

There exist several frameworks for the evaluation of supervised tasks involving GNNs, but none for unsupervised tasks such as community detection. Palowitch et al. (2022) provides an empirical study of the performance of 11 GNNs that are representative of the research space as well as introducing a novel methodology for generating synthetic graphs. They assess performance in node classification, link prediction and feature prediction. This helps to assess how GNNs might perform in practical, untested applications. They provide multiple attempts at fair hyperparameter optimisation across the models. These modes of optimisation specified are randomly drawing hyperparameters or evaluating a series of the randomly draw hyperparameters against some evaluation metric. They illustrate the usefulness of empirical evaluation and that a consistent framework across models is the best way to assess progress. Errica et al. (2019) also provides a reproducible and rigorous environment for evaluating graph classification. The authors use this setup to justify that GNNs are not yet fully exploiting structural information present, reasoning that previous comparisons are subject to dependence of random initialisations.

There exists recent studies also showing that fair GNN evaluations are required to accurately compare methods; Huang et al. (2020) demonstrates that simpler models can outperform GNNs when evaluated with a proper baseline, highlighting a gap in research knowledge between community detection and GNN research, which we seek to address. Shchur et al. (2018) show that there are various shortcomings in the evaluation of GNNs where they often lack a fair comparison for different architectures. Importantly, they show that under similar testing conditions, simple architectures can perform better than complex. They empathise the need for evaluation strategies based on multiple splits, due to the unstable nature and the high dependence of model performance on training environment. Chen et al. (2022) performs an experimental evaluation of the tricks that various GNN methods use to improve their methods but applied to a baseline GNN. They find that there are sweet spots of architecture trick combinations for each dataset that give the best performance, providing motivation for the exploration of the potential combination in hyperparameter space.

The performance of a method depends on the dataset used in evaluation (Palowitch et al., 2022). Hu et al. (2020) presents large scale graph benchmarks across a range of domains and tasks. They use data splits that reflect practical usage by partitioning the graph based on a metric of node importance. Generally, there are a wide range of benchmarks for graph learning Dwivedi et al. (2020); Morris et al. (2020); Liu et al. (2022) but currently none for community detection.

3 Methods

The relevant research thread for methodologies that learn representations with GNNs is summarised in Section 3.1. Thereafter, we group GNN methods surveyed into two families: unsupervised graph clustering (Section 3.2) and self-supervised graph learning (Section 3.3). The former refers to GNN architectures designed with the modules that are intended for the purpose of detecting communities, the latter use contrastive learning schemes for representation learning. Typically, contrastive methods are evaluated using logistic regression on the representation learnt by the GNN, as they have no functionality to cluster. Therefore, instead, we apply k-means to the learnt representations of the graph in vector space to detect communities unsupervised clustering results.

3.1 Graph Representation Learning

GNNs process a graph by computing a function (in the form of a neural network) over both structural connectivity and feature information. GNNs were first formulated by Scarselli et al. (2008) to combine graph data with node information by propagating feature information messages across a node's connectivity. The general framework of aggregating information from a node's neighbourhood into an embedding was defined as Message Passing Neural Networks by Gilmer et al. (2017). Graph Convolutional Networks (GCNs) expands on the GNN formulation by using a learn-able weight matrix through which messages are passed. To efficiently compute the averaging of neighbourhood features a symmetric normalisation of the Laplacian is used in combination with a re-normalisation trick to solve vanishing gradients. Graph Attentional Networks (GATs) propose that each node should be able to individually learn the weights of contribution from their direct neighbours Veličković et al. (2017). GAT makes it possible to assign varying importance's to different nodes within a neighborhood whilst also considering different sized neighbourhoods. The following methods will use these representations produced by a GNN for community detection.

3.2 Unsupervised Graph Clustering

GNN methods detailed herein were proposed to specifically use a clustering objective, formulated in various ways, while training the neural network. The **Deep Attentional Embedded Graph Clustering (DAEGC)** Wang et al. (2019a) architecture consists of an encoder variation of a GAT, that exploits high-order neighbors by use of a proximity matrix, and a inner product decoder trained a reconstruction loss to recover graph structure. They extends ideas from Xie et al. (2016) by using k-means target to self-supervise the clustering module to iteratively refine the clustering of node embeddings. **Deep Modularity Networks (DMoN)** is a method for attributed graph clustering with GCNs by maximising a modularity based clustering objective to optimise cluster assignments by a spectral relaxation of the problem Tsitsulin et al. (2020). **Variational Graph AutoEncoder Reconstruction (VGAER)** reconstructs a modularity distribution using a cross entropy based decoder from the encoding of a VGAE Kipf & Welling (2016b).

3.3 Self-Supervised Graph Learning

Self-supervision is often used in GNNs, and in particular, the idea of contrastive learning. Contrastive methods compare samples of data with the intuition that similar samples in the original domain should be embedded close to each other in the embedded domain. This process typically employs the use of augmentation to produce a sample that has preserved information pertinent to the representation. **Deep Graph Infomax (DGI)** adapts principles from Deep InfoMax Hjelm et al. (2018) for graph applications by maximising mutual information between patch representations of sub-graphs and the corresponding high-level summaries Velickovic et al. (2019). This helps the model to contain information that is globally relevant by encouraging information to be preserved that is present across the local patch summaries. **GRAph Contrastive rEpresentation learning (GRACE)** generates a corrupted view of the graph by removing edges and masking features and learns node representations by maximising agreement across the viewsZhu et al. (2020). They argue that the mean-pooling readout function used by DGI to summarise the graph is insufficient to preserve distinctive features from node-level embeddings Zhu et al. (2020). **Contrastive Multi-View Representation Learning on Graphs (MVGRL)** argues that the best employment of contrastive methods for graphs is achieved by contrasting encodings' from first-order neighbors and a general graph diffusion Hassani & Khasahmadi (2020). By use of a mutual information estimator the agreement between representations from both sides of the architecture is scored. **Bootstrapped Graph Latents (BGRL)** Thakoor (2021) adapts the self-supervised bootstrap procedure from BYOL Grill et al. (2020), by maintaining two graph encoders. The online encoder learns to predict the representations of the target encoder, which in itself is updated by an exponential moving average of the online encoder. **SelfGNN** Kefato & Girdzijauskas (2021) also uses this principal but uses augmentations of the feature space to train the network.

4 Framework

4.1 Problem Definition

The problem definition of community detection on attributed graphs is defined as follows. The graph is represented as $G = (A, X)$, with the relational information of nodes modelled by the adjacency matrix $A \in \mathbb{R}^{N \times N}$. Given a set of nodes V and a set of edges E, let $e_{i,j} = (v_i, v_j) \in E$ denote the edge that points from v_j to v_i. The graph is considered weighted so, the adjacency matrix $0 < A_{i,j} \leq 1$ if $e_{i,j} \in E$ and $A_{i,j} = 0$ if $e_{i,j} \notin E$. Also given is a set of node features $X \in \mathbb{R}^{N \times d}$, where N is the number of nodes in the graph and d represents the number of different node attributes (or feature dimensions). The objective is to partition the graph G into k clusters such that nodes in each partition, or cluster, generally have similar structure and feature values. For simplicity we assume hard clustering, where each community detection algorithm must predict for each nodes a single community to which it belongs, such that $P \in \mathbb{R}^N$. Community detection using traditional methods requires the intermediate step of producing a list of edges to split in order to form communities based on the adjacency matrix. This is not required in our framework as we will only evaluate the clusters associated with each node using the labels given such that $L \in \mathbb{R}^N$.

4.2 DATASETS

It has recently been further demonstrated (Palowitch et al., 2022) that the selection of datasets affects the ranking of different methods at the related task of node classification. Building on this, we compare methods across a range of graph topologies as defined by their graph statistics. To summarise each dataset in terms of structural clusters, or communities, we use the average clustering coefficient which is simply the average of all the local clustering coefficients calculated for each node, which is the proportion of all the connections that exist in a nodes neighbourhood, over all the links that could exist in that neighbourhood (Watts & Strogatz, 1998). We also use the average closeness centrality as a measure of how close the average node is to other nodes. Closeness centrality is defined as the reciprocal of the mean shortest path distance from all other nodes in the graph (Wasserman et al., 1994). We use publicly available datasets that have been used previously in GNN research, shown along with their statistics available Table 1. A more detailed description of each dataset can be found in the appendix.

Table 1: The datasets, and associated statistics, we will study.

Datasets	Nodes	Edges	Features	Classes	Average Clustering Coefficient	Mean Closeness Centrality
acm (Wang et al., 2019b)	3025	3025	13128	3	0.549	0.079
amac (He & McAuley, 2016)	13752	13752	80062	10	0.157	0.264
amap (He & McAuley, 2016)	7650	7650	119081	8	0.404	0.242
bat	131	131	1038	4	0.636	0.469
citeseer (Giles et al., 1998)	3327	3327	4552	6	0.141	0.045
cora (McCallum et al., 2000)	2708	2708	5278	7	0.241	0.137
dblp (Tang et al., 2008)	4057	4057	3528	4	0.177	0.026
eat	399	399	5994	4	0.539	0.441
uat	1190	1190	13599	4	0.501	0.332
texas (Craven et al., 1998)	183	183	162	5	0.198	0.344
wisc (Craven et al., 1998)	251	251	257	5	0.208	0.32
cornell (Craven et al., 1998)	183	183	149	5	0.167	0.326

4.3 PERFORMANCE METRICS

There exist several different performance metrics commonly used to evaluate community detection, including metrics that assume access to the ground truth, and others that do not. For the former, we use the (macro) F1-score, and the normalised mutual information (NMI). We use the macro F1-score (Van Rijsbergen, 1977) to ensure that our evaluations are not biased by the label distribution, as communities sizes are often imbalanced. Macro-F1 (Lewis et al., 1996) is computed using an unweighted mean of all the per-class F1 scores. NMI is a widely used measure to compare community detection methods. NMI is the normalisation of the amount of information that can extract from one distribution with respect to a second (Lancichinetti et al., 2009). These two metrics are calculated using the information of the true cluster labels, and as such can be considered supervised metrics, although this knowledge is not used during the community detection process itself. Modularity (Newman, 2006) quantifies the deviation of the clustering from what would be observed in expectation under a random graph (Tsitsulin et al., 2020). Conductance (Shi & Malik, 2000) is the proportion of total edge volume that points outside the cluster which Yang & Leskovec (2012) shows is a good measure of evaluating network communities. These two metrics are unsupervised as they are calculated using the predicted cluster label and the adjacency matrix, without using a ground truth.

4.4 Evaluation Environment

All community detection methods have hyperparameters that influence the performance of the method. Unfortunately, without a consistent framework, particularly in the unsupervised setting, it is often unclear in the literature how hyperparameters were found. To the best of our knowledge, this holds for all GNN based methods we cover here. Soenen et al. (2021) details the effects of not choosing hyperparameters by starting with the assumption that there is no best way of selecting hyperparameters. They first select those given in their respective original papers however, the default settings can be selected based on observations not known or detailed in the original paper; this can bias algorithms performance to the original datasets. They then use a selection method called 'best default' which means to compute the best defaults given access to some datasets and then use those found values on new datasets. Selection of hyperparameters in this way assumes that a researcher will always have knowledge of external sources in order to select the best performance. The main conclusion must be that newly proposed methods must have a procedure to compare to, otherwise they are using potentially different information to evaluate their choices of parameters, calling into question the fairness of novel investigations. These conclusions were drawn within the context of anomaly detection, which we seek to build on here within community detection.

As it is well known that the best hyperparameters for a particular method are dataset dependent, we will use hyperparameter optimisation (HPO) to evaluate each method with rigour. We consider best practice for methods is to test across a wide a range of HPs, with as wide a HP interval as a budget allows. Choosing too wide of a HP interval or including uninformative HPs in the search space can have an adverse effect on tuning outcomes in the given budget Bischl et al. (2021); Yang & Shami (2020). The hyperparameter we measure can be found in the Appendix. We note that it is not our objective to discover the *best* HPs for each method across each dataset, thus we will perform HPO under feasible constraints in order to validate our hypothesis.

For HPO, for each dataset, we train the GNN based method on a subset of the data, as done in supervised machine learning, and select hyperparameters based on a performance metric on the validation set. Given that the Tree Parzen-Estimator (TPE) Bergstra et al. (2013) can retain the conditionality of variables Yang & Shami (2020) and has been shown to be a good estimator given limited resources Yuan et al. (2021), we will use the TPE for this evaluation environment.

For a fair comparison, all GNNs methods tested will be trained using the Adam optimiser Kingma & Ba (2014) but will be treat the learning rate and decay as HPs. All methods may train for a maximum amount of epochs (5000), but we implement a patience criteria for early stopping. We allocate a total max budget 200000 choices for each algorithm and a budget of 250 rounds of training. Algorithms that are designed to benefit from a small number of HPs should perform better than those with more, as they can search more of the space within the given budget. The full hyperparameter space explored is detailed in Appendix A. We split the dataset by removing edges from the adjacency matrix with $1 - p$ (where p is 0.8 in our evaluations). The full experiment procedure is given by Algorithm 1. Using this procedure, we answer the research question: to what extent are the ranking of each method affected by a consistent experimental evaluation procedure?

5 Evaluation

In this section we will evaluate a variety of methods under our proposed framework, analyse and discuss their results. At a high level, we find, as recently shown by Palowitch et al. (2022) in another GNN task, that the best method is not ubiquitously the best across all datasets. Similarly, the worst method on each dataset is not the same. Further, how the *best* or *worst* is measured depends on the specific metric used to quantify performance.

From the results in Figure 1, we observe on every dataset tested that the performance of at least one model is improved due to the use of HPO in our framework, beyond the originally reported hyperparameters. While the origin of how the originally reported hyperparameters were found is often unclear, it is evident that reported hyperparameters should be found using a consistent procedure as the GNN based approaches are clearly sensitive to them for community detection.

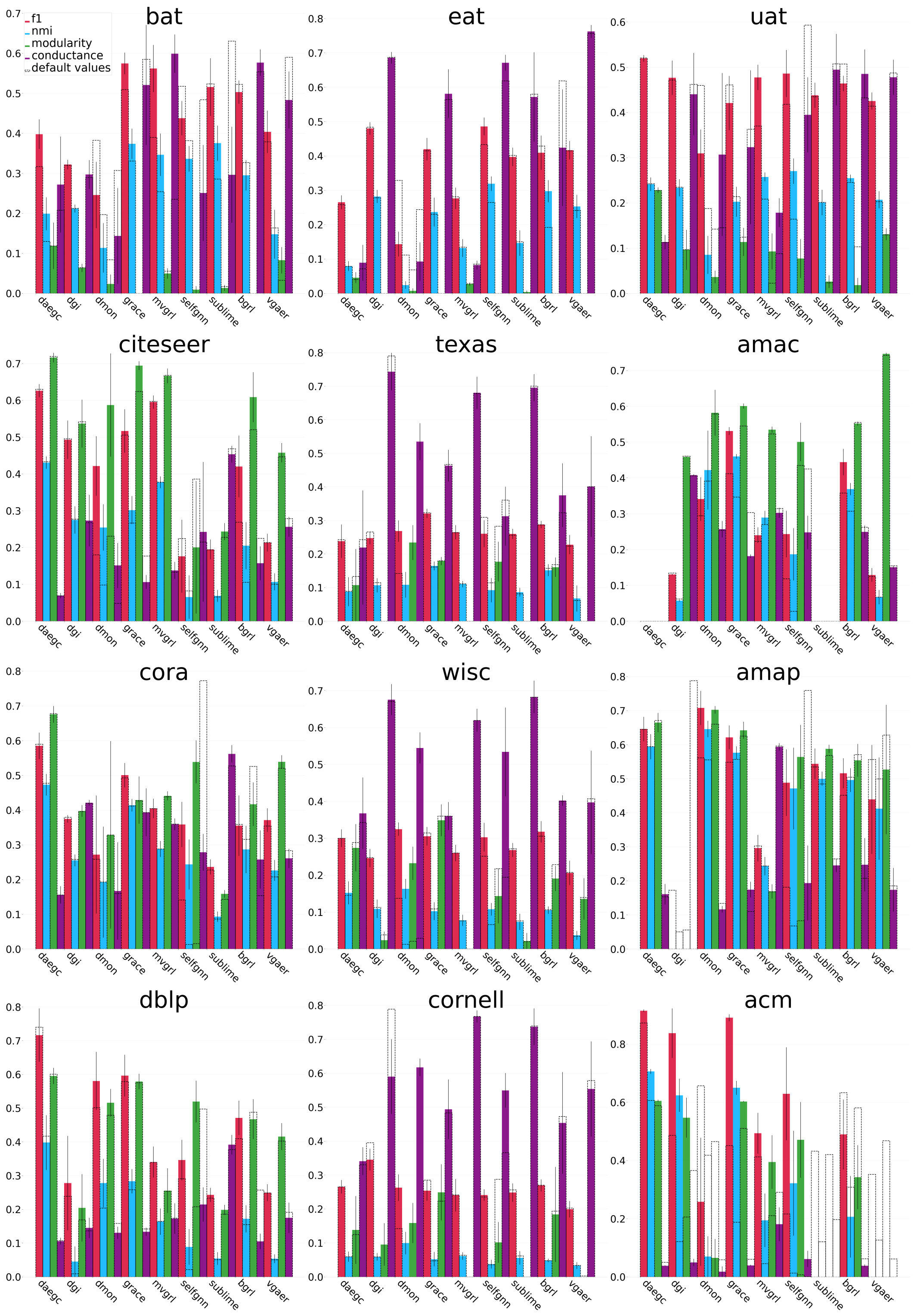

Figure 1: The average performance and standard deviation of all metrics over all methods and datasets, from the hyperparameter investigation (shown in colour), compared with the default hyperparameters (dashed bars). The metric maximised in the HPO was NMI. Conductance is the only metric where a lower value is better. Out Of Memory (OOM) occurrences during the HPO were the following dataset/method combinations: amac-daegc, amac-sublime, amap-dgi, acm-sublime, acm-vgaer; the dataset/method combinations that caused OOM using the default settings were amac-daegc, amac-sublime. All models are trained on a server with 4x 2080 Ti GPUs (12GB), 96GB of RAM, and a 16core Xeon CPU.

Algorithm 1 Evaluation Environment

```
1:  for seed_idx $= 1, 2, \ldots, 10$ do
2:      train_dataset/test_dataset = split(dataset, $p = 0.8$)
3:      if seed_idx $= 1$ then
4:          best_performance = 0.
5:          training_dataset/validation-dataset = split(train_dataset, $p = 0.8$)
6:          for n_trial $= 1, 2, \ldots, 250$ do
7:              hparams $\sim$ TPE
8:              model.train(hparams, training_dataset)
9:              performance = model.eval(validation_dataset)
10:             if performance $>$ best_performance then
11:                 best_performance = performance, best_hparams = hparams
12:             end if
13:         end for
14:     end if
15:     model.train(best_hparams, train_dataset)
16:     test_performance[seed_idx] = model.eval(test_dataset)
17: end for
18: final_performance $= \sum$ test_performance / 10
```

The HPO can be seen to influence the F1 performance ranking on most datasets, except for cora. This is notable because cora is the most commonly used dataset in general GNN research, indicating a potential bias towards this dataset in measuring performance. From this we can also reason that there may be a bias of GNN architecture design towards certain graph data topologies, which can be addressed by using a wide and consistent evaluation framework. If we look at the acm dataset, the discovered hyperparameters perform worse than the original parameters across the methods tested. This indicates that methods in general perform worse on this graph topology. The acm dataset is the only dataset that has a high average clustering coefficient combined with a low closeness centrality. From this, we may infer that GNN based community detection doesn't work well when clusters are densely connected but the average paths between nodes are long.

In general, F1 and NMI appear to be correlated in terms of ranking between default and HPO hyperparameters. However, if we look across the datasets, the conductance and modularity are often vastly different between the two sets of hyperparameters despite the similarity in the other metrics. This highlights the importance of choosing the optimisation target for HPO, with our analysis produced by optimising over a supervised NMI metric on the validation set.

Further, the results demonstrate that each method on each dataset is sensitive to the random seeds used. It is unsurprising that there is deviation due to the random seed, but it is clear the necessity to report results in this way for community detection, as the random seed impacts not only the methods themselves, but also the splits used for training, validation and testing.

The difference between the results of the HPO and the default parameters specified, show that as expected, results are significantly affected by the evaluation procedure. We propose that future research makes use of our framework, or similar, in order to ensure that all community detection methods are evaluated under the same procedure in order to increase fairness and rigour. We show that if new methods merely utilise hyperparameters for methods as reported in their original research papers, it can be difficult to ensure the rigour needed, given the sensitivity of methods to them and the lack of clarity in how they were found. Given the unsupervised nature of community detection, along with the common use of deep neural networks, there are less established protocols for setting hyperparameters, and it may be unclear the degree of 'overfitting' to datasets that took place to find hyperparameters.

Our results also demonstrate that there is currently no consistent State-of-the-Art method in community detection across even the limited set of common datasets we used for our evaluation, when using a consistent evaluation framework.

5.1 LIMITATIONS

We readily acknowledge a number of limitations of our study. Firstly, we evaluate over a limited search space, which could be extended to a continuous distribution over each parameter, with more trials and thus more computational time. However, we chose a search space that is feasible for typical researchers to search over, and thus, practical and usable.

In this paper, we evaluate the learnt embedding dimension of contrastive methods with k-means but do not compare performance using other clustering methods. It may be relevant to include spectral clustering or density based clustering algorithms for comparison. Nonetheless, k-means is well used in the literature for clustering deep neural network representations, and thus is a reasonable choice.

6 CONCLUSION

In this work we sought to propose a solution to a challenging problem within the nascent field of community detection with GNNs. We outlined the reasons why it is often difficult to compare methods currently at this unsupervised task and proposed a framework for the fair and consistent evaluation of them. Using this framework we evaluated a suite of community detection algorithms across many benchmark graph datasets used for community detection. Our findings confirm that all current methods are sensitive to the experimental evaluation procedure used, the details of which are often missing from the respective papers. We hope that this work establishes a common protocol to help the field of GNN based community detection to grow and flourish.

REFERENCES

James Bergstra, Daniel Yamins, and David Cox. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In *International conference on machine learning*, pp. 115–123. PMLR, 2013.

Bernd Bischl, Martin Binder, Michel Lang, Tobias Pielok, Jakob Richter, Stefan Coors, Janek Thomas, Theresa Ullmann, Marc Becker, Anne-Laure Boulesteix, et al. Hyperparameter optimization: Foundations, algorithms, best practices and open challenges. *arXiv preprint arXiv:2107.05847*, 2021.

Irineo Cabreros, Emmanuel Abbe, and Aristotelis Tsirigos. Detecting community structures in hi-c genomic data. In *2016 Annual Conference on Information Science and Systems (CISS)*, pp. 584–589. IEEE, 2016.

P Chen and Sidney Redner. Community structure of the physical review citation network. *Journal of Informetrics*, 4(3):278–290, 2010.

Tianlong Chen, Kaixiong Zhou, Keyu Duan, Wenqing Zheng, Peihao Wang, Xia Hu, and Zhangyang Wang. Bag of tricks for training deeper graph neural networks: A comprehensive benchmark study. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2022.

Petr Chunaev. Community detection in node-attributed social networks: a survey. *Computer Science Review*, 37:100286, 2020.

Mark Craven, Andrew McCallum, Dan PiPasquo, Tom Mitchell, and Dayne Freitag. Learning to extract symbolic knowledge from the world wide web. Technical report, Carnegie-mellon univ pittsburgh pa school of computer Science, 1998.

Vijay Prakash Dwivedi, Chaitanya K Joshi, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Jubin Edachery, Arunabha Sen, and Franz J Brandenburg. Graph clustering using distance-k cliques. In *International Symposium on Graph Drawing*, pp. 98–106. Springer, 1999.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. *arXiv preprint arXiv:1912.09893*, 2019.

Absalom E Ezugwu, Abiodun M Ikotun, Olaide O Oyelade, Laith Abualigah, Jeffery O Agushaka, Christopher I Eke, and Andronicus A Akinyelu. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Engineering Applications of Artificial Intelligence*, 110:104743, 2022.

C Lee Giles, Kurt D Bollacker, and Steve Lawrence. Citeseer: An automatic citation indexing system. In *Proceedings of the third ACM conference on Digital libraries*, pp. 89–98, 1998.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *International conference on machine learning*, pp. 1263–1272. PMLR, 2017.

Jean-Bastien Grill, Florian Strub, Florent Altché, Corentin Tallec, Pierre H Richemond, Elena Buchatskaya, Carl Doersch, Bernardo Avila Pires, Zhaohan Daniel Guo, Mohammad Gheshlaghi Azar, et al. Bootstrap your own latent: A new approach to self-supervised learning. *arXiv preprint arXiv:2006.07733*, 2020.

Kaveh Hassani and Amir Hosein Khasahmadi. Contrastive multi-view representation learning on graphs. In *International Conference on Machine Learning*, pp. 4116–4126. PMLR, 2020.

Ruining He and Julian McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *proceedings of the 25th international conference on world wide web*, pp. 507–517, 2016.

R Devon Hjelm, Alex Fedorov, Samuel Lavoie-Marchildon, Karan Grewal, Phil Bachman, Adam Trischler, and Yoshua Bengio. Learning deep representations by mutual information estimation and maximization. *arXiv preprint arXiv:1808.06670*, 2018.

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.

Qian Huang, Horace He, Abhay Singh, Ser-Nam Lim, and Austin R Benson. Combining label propagation and simple models out-performs graph neural networks. *arXiv preprint arXiv:2010.13993*, 2020.

Di Jin, Zhizhi Yu, Pengfei Jiao, Shirui Pan, Dongxiao He, Jia Wu, Philip Yu, and Weixiong Zhang. A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Transactions on Knowledge and Data Engineering*, 2021.

Zekarias T Kefato and Sarunas Girdzijauskas. Selfgnn: Self-supervised graph neural networks without explicit negative sampling. 2021.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Andrea Lancichinetti, Santo Fortunato, and János Kertész. Detecting the overlapping and hierarchical community structure in complex networks. *New journal of physics*, 11(3):033015, 2009.

David D Lewis, Robert E Schapire, James P Callan, and Ron Papka. Training algorithms for linear text classifiers. In *Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval*, pp. 298–306, 1996.

Fanzhen Liu, Shan Xue, Jia Wu, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Jian Yang, and Philip S Yu. Deep learning for community detection: progress, challenges and opportunities. *arXiv preprint arXiv:2005.08225*, 2020.

Kay Liu, Yingtong Dou, Yue Zhao, Xueying Ding, Xiyang Hu, Ruitong Zhang, Kaize Ding, Canyu Chen, Hao Peng, Kai Shu, et al. Benchmarking node outlier detection on graphs. *arXiv preprint arXiv:2206.10071*, 2022.

Andrew Kachites McCallum, Kamal Nigam, Jason Rennie, and Kristie Seymore. Automating the construction of internet portals with machine learning. *Information Retrieval*, 3(2):127–163, 2000.

Federico Monti, Fabrizio Frasca, Davide Eynard, Damon Mannion, and Michael M Bronstein. Fake news detection on social media using geometric deep learning. *arXiv preprint arXiv:1902.06673*, 2019.

Atefeh Moradan, Andrew Draganov, Davide Mottin, and Ira Assent. Ucode: Unified community detection with graph convolutional networks. *arXiv preprint arXiv:2112.14822*, 2021.

Christopher Morris, Nils M Kriege, Franka Bause, Kristian Kersting, Petra Mutzel, and Marion Neumann. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv preprint arXiv:2007.08663*, 2020.

Mark EJ Newman. Modularity and community structure in networks. *Proceedings of the national academy of sciences*, 103(23):8577–8582, 2006.

John Palowitch, Anton Tsitsulin, Brandon Mayer, and Bryan Perozzi. Graphworld: Fake graphs bring real insights for gnns. *arXiv preprint arXiv:2203.00112*, 2022.

Guillaume Salha-Galvan, Johannes F Lutzeyer, George Dasoulas, Romain Hennequin, and Michalis Vazirgiannis. Modularity-aware graph autoencoders for joint community detection and link prediction. *arXiv preprint arXiv:2202.00961*, 2022.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE transactions on neural networks*, 20(1):61–80, 2008.

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

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

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

Jonas Soenen, Elia Van Wolputte, Lorenzo Perini, Vincent Vercruyssen, Wannes Meert, Jesse Davis, and Hendrik Blockeel. The effect of hyperparameter tuning on the comparative evaluation of unsupervised anomaly detection methods. In *Proceedings of the KDD'21 Workshop on Outlier Detection and Description*, pp. 1–9. Outlier Detection and Description Organising Committee, 2021.

Xing Su, Shan Xue, Fanzhen Liu, Jia Wu, Jian Yang, Chuan Zhou, Wenbin Hu, Cecile Paris, Surya Nepal, Di Jin, et al. A comprehensive survey on community detection with deep learning. *IEEE Transactions on Neural Networks and Learning Systems*, 2022.

Jie Tang, Jing Zhang, Limin Yao, Juanzi Li, Li Zhang, and Zhong Su. Arnetminer: extraction and mining of academic social networks. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 990–998, 2008.

Tallec C. Azar M. G. Munos R. Veličković P. Valko M. Thakoor, S. Bootstrapped representation learning on graphs. *ICLR Workshop on Geometrical and Topological Representation Learning*, 2021.

Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. Graph clustering with graph neural networks. *arXiv preprint arXiv:2006.16904*, 2020.

Cunchao Tu, Xiangkai Zeng, Hao Wang, Zhengyan Zhang, Zhiyuan Liu, Maosong Sun, Bo Zhang, and Leyu Lin. A unified framework for community detection and network representation learning. *IEEE Transactions on Knowledge and Data Engineering*, 31(6):1051–1065, 2018.

Cornelis Joost Van Rijsbergen. A theoretical basis for the use of co-occurrence data in information retrieval. *Journal of documentation*, 1977.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. *arXiv preprint arXiv:1710.10903*, 2017.

Petar Velickovic, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. *ICLR (Poster)*, 2(3):4, 2019.

Chun Wang, Shirui Pan, Ruiqi Hu, Guodong Long, Jing Jiang, and Chengqi Zhang. Attributed graph clustering: A deep attentional embedding approach. *arXiv preprint arXiv:1906.06532*, 2019a.

Xiao Wang, Houye Ji, Chuan Shi, Bai Wang, Yanfang Ye, Peng Cui, and Philip S Yu. Heterogeneous graph attention network. In *The world wide web conference*, pp. 2022–2032, 2019b.

Stanley Wasserman, Katherine Faust, et al. Social network analysis: Methods and applications. 1994.

Duncan J Watts and Steven H Strogatz. Collective dynamics of 'small-world'networks. *nature*, 393 (6684):440–442, 1998.

Junyuan Xie, Ross Girshick, and Ali Farhadi. Unsupervised deep embedding for clustering analysis. In *International conference on machine learning*, pp. 478–487. PMLR, 2016.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. In *Proceedings of the ACM SIGKDD Workshop on Mining Data Semantics*, pp. 1–8, 2012.

Jaewon Yang, Julian McAuley, and Jure Leskovec. Community detection in networks with node attributes. In *2013 IEEE 13th international conference on data mining*, pp. 1151–1156. IEEE, 2013.

Li Yang and Abdallah Shami. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing*, 415:295–316, 2020.

Yingfang Yuan, Wenjun Wang, and Wei Pang. A systematic comparison study on hyperparameter optimisation of graph neural networks for molecular property prediction. In *Proceedings of the Genetic and Evolutionary Computation Conference*, pp. 386–394, 2021.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Deep graph contrastive representation learning. *arXiv preprint arXiv:2006.04131*, 2020.

Yanqiao Zhu, Yichen Xu, Feng Yu, Qiang Liu, Shu Wu, and Liang Wang. Graph contrastive learning with adaptive augmentation. In *Proceedings of the Web Conference 2021*, pp. 2069–2080, 2021.

A Appendix

A.1 Dataset Description

Cora was introduced by McCallum et al. (2000) and represents a graph of papers in topics around machine learning, where each publication is represented by a Bag of Words model. **Citeseer** Giles et al. (1998) was created from publications relating to neural neural, and the feature matrix is a TF/IDF weighted word vector. The **Acm** dataset contains papers published KDD, SIGMOD, SIGCOMM, MobiCOM and are divided into three clusters by research area - databases, wireless communication, data mining Wang et al. (2019b). The feature matrix comes from a bag-of-words of the keywords from the paper. The DBLP is a citation network dataset extracted from DBLP, ACM, MAG (Microsoft Academic Graph), and other sources Tang et al. (2008).

Amac and **Amap** are extracted from the Amazon co-purchase graph (He & McAuley, 2016), where nodes represent products, edges represent whether two products are frequently co-purchased or not, features represent product reviews encoded by bag-of-words. **Texas, Wisc,** and **Cornell** are extracted from the WebKB dataset that includes web pages from computer science departments of various universities (Craven et al., 1998). **Uat, Eat,** and **Bat** contain airport activity data collected from different sources. Uat is extracted from National Civil Aviation Agency (ANAC) from January to December 2016 and looks at the number of people passing through each airport. Eat and Bat features are based on number of landings are are respectively extracted from Statistical Office of the European Union (Eurostat) and Bureau of Transportation Statistics.

A.2 Methods not explored due to constraints with processing power time

Salha-Galvan et al. (2022) is similar to VGAER with a slight architectural change calling their model the Modularity-Aware VGA. They using a s-regular specification of the adjacency matrix, which connects nodes randomly rather than considering the connection to all nodes as done with VGAER.

Graph Contrastive Learning with Adaptive Augmentation (GCA) Zhu et al. (2021) uses the same architecture as GRACE but argues that data augmentation schemes do not do enough to preserve the intrinsic structures and attributes of graphs. They suggest altering masking probabilities to be skewed for unimportant edges or features, so that important features are more likely not be masked. Again, due to time constraints, this method will not be evaluated in this study.

Unified Community Detection with Graph Convolutional Networks (UCoDe) detects communities by comparing node similarity on a community-wise modularity matrix Moradan et al. (2021). This architecture uses a GCN to output node community affiliation probabilities from which a modularity matrix is constructed. A corrupted view of this matrix is constructed by a row-wise permutation of the matrix then compared with the original in the loss function. This has the effect of maximising the diagonal elements of the modularity matrix which represents the modularity within a community and minimising the off-diagonal elements.

A.3 hyperparameter configuration

cross model parameters

patience : [25, 100, 500, 1000]

learning rate : [0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001]

weight decay : [0.05, 0.005, 0.0005, 0.0]

size of **search space: 96**

daegc

pre epoch : [30, 40, 50, 60]

pre lr : [0.01, 0.005, 0.001]

update interval : [1, 3, 5]

hidden size : [128, 256, 512]

embedding size : [8, 16, 24]

kl loss const : [1, 0.2, 5, 10, 20]

size of **search space: 155520**

dgi

hid units : [64, 128, 256, 512]

size of **search space: 384**

dmon

architecture : [32, 64, 128, 256, 512]

cluster size regularization : [0, 0.3, 0.5, 0.7, 1.0]

dropout rate : [0.75, 0.5, 0.25, 0.1]

size of **search space: 9600**

grace

num hidden : [128, 256]
num proj hidden : [64, 128]
drop edge rate 1 : [0.1, 0.2, 0.4]
drop edge rate 2 : [0.1, 0.2, 0.4]
drop feature rate 1 : [0.0, 0.1, 0.2]
drop feature rate 2 : [0.0, 0.1, 0.2]
size of **search space: 31104**

mvgrl

hid units : [64, 128, 256, 512]
sample size : [1500, 2000]
batch size : [1, 2, 4]
size of **search space: 2304**

selfgnn

gnn type : [’gcn’, ’gat’, ’sage’]
prd head norm : [’batch’, False]
prj head norm : [’layer’, False]
encoder norm : [’no’, False]
layer1 : [512, 256]
layer2 : [64, 128, 256]
dropout : [0, 0.25, 0.5]
size of **search space: 41472**

sublime

hidden dim : [256, 512]
rep dim : [32, 64]
proj dim : [32, 64]
dropout : [0, 0.5]
maskfeat rate learner : [0.2, 0.5, 0.8]
maskfeat rate anchor : [0.2, 0.5, 0.8]
dropedge rate : [0.25, 0.5, 0.75]
k : [20, 30, 40]
size of **search space: 124416**

bgrl

layer1 : [512, 256]
layer2 : [128, 256]
drop feature rate 1 : [0.1, 0.2, 0.4]
drop feature rate 2 : [0.1, 0.2, 0.4]
drop edge rate 1 : [0.1, 0.2, 0.4]
drop edge rate 2 : [0.1, 0.2, 0.4]
dropout : [0.0, 0.25, 0.5]

size of **search space: 93312**

vgaer

hidden1 : [16, 32, 64, 128]
hidden2 : [8, 16, 32, 64]
dropout : [0.0, 0.2, 0.5]
size of **search space: 4608**