

## Article

# Uncertainty in GNN Learning Evaluations: A Comparison between Measures for Quantifying Randomness in GNN Community Detection <sup>†</sup>

William Leeney <sup>\*</sup> and Ryan McConville 

School of Engineering Mathematics and Technology, University of Bristol, Bristol BS8 1TR, UK;  
ryan.mcconville@bristol.ac.uk

<sup>\*</sup> Correspondence: will.leeney@bristol.ac.uk

<sup>†</sup> This paper is an extended version of our paper published in The 12th International Conference on Complex Networks and their Applications (2023).

**Abstract:** (1) The enhanced capability of graph neural networks (GNNs) in unsupervised community detection of clustered nodes is attributed to their capacity to encode both the connectivity and feature information spaces of graphs. The identification of latent communities holds practical significance in various domains, from social networks to genomics. Current real-world performance benchmarks are perplexing due to the multitude of decisions influencing GNN evaluations for this task. (2) Three metrics are compared to assess the consistency of algorithm rankings in the presence of randomness. The consistency and quality of performance between the results under a hyperparameter optimisation with the default hyperparameters is evaluated. (3) The results compare hyperparameter optimisation with default hyperparameters, revealing a significant performance loss when neglecting hyperparameter investigation. A comparison of metrics indicates that ties in ranks can substantially alter the quantification of randomness. (4) Ensuring adherence to the same evaluation criteria may result in notable differences in the reported performance of methods for this task. The  $W$  randomness coefficient, based on the Wasserstein distance, is identified as providing the most robust assessment of randomness.



**Citation:** Leeney, W.; McConville, R. Uncertainty in GNN Learning Evaluations: A Comparison between Measures for Quantifying Randomness in GNN Community Detection. *Entropy* **2024**, *26*, 78.

<https://doi.org/10.3390/e26010078>

Academic Editor: Hocine Cherifi

Received: 14 December 2023

Revised: 4 January 2024

Accepted: 15 January 2024

Published: 17 January 2024



**Copyright:** © 2024 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (<https://creativecommons.org/licenses/by/4.0/>).

## 1. Introduction

Graph neural networks (GNNs) have gained popularity as a neural network-based approach for handling graph-structured data, leveraging their capacity to merge two information sources through the propagation and aggregation of node feature encodings along the network's connectivity [1]. Nodes within a network can be organised into communities based on similarities in associated features and/or edge density [2]. Analyzing the network structure to identify clusters or communities of nodes proves valuable in addressing real-world issues like misinformation detection [3], genomic feature discovery [4], and social network or research recommendation [5]. We consider unsupervised neural approaches to community detection that do not use any ground truth or labels during training to optimise the loss functions. As an unsupervised task, the identification of node clusters relies on latent patterns within the dataset rather than on "ground-truth" labels. Clustering holds significance for emerging applications lacking associated ground truth. Evaluating performance in discovering unknown information becomes crucial for applications where label access is restricted. Many graph applications involve millions of nodes, and datasets mimicking realistic scenarios exhibit low labeling rates [6].

Recently, Leeney and McConville [7] proposed a framework for fairly evaluating GNN community detection algorithms and the importance of a hyperparameter optimisation

procedure to performance comparisons. However, this is not done consistently across the field, although benchmarks are widely considered as important. Confusion arising from biased benchmarks and evaluation procedures distorts understanding of the research field. Currently, resources, money, time, and energy training models are wasted on inconclusive results which may have developed understanding in the field but go unpublished. Where research findings inform policy decisions or medical practices, publication bias can lead to decisions based on incomplete or biased evidence, in turn creating inefficiencies and downstream harms. To accurately reflect the real-world capabilities of research, one may use a common framework for evaluating proposed methods. To demonstrate the need for such a framework in this setting, we measure the difference between using the default parameters given by the original implementations to those optimised for under this framework. To quantify the influence of randomness on results, we compare three metrics, two proposed herein, for evaluating the consistency of algorithm rankings over different random seeds which quantifies the robustness of results. Here, we also investigate how the quantification of randomness is affected when ties in performance between two or more algorithms is accounted for. Ties are can occur when out-of-memory errors occur or when a metric such as conductance is optimised. In addition, we investigate how these metrics change as the number of tests increase, showing how sensitive each is to the breadth of investigations.

### Related Work

There is a recognition of the necessity for rigor in frameworks assessing machine learning algorithms [8]. Several frameworks for assessing the performance of supervised GNNs performing node classification, link prediction and graph classification exist [9–12]. Our focus lies in unsupervised community detection, a task that proves more challenging to train and evaluate. While existing reviews on community detection offer insights, they generally lack thorough evaluations. The absence of a standardised evaluation in this task has been acknowledged in discussions concerning non-neural methods [13,14], but this consideration does not currently extend to GNNs.

Su et al. [15] provide a taxonomy of existing node clustering methods, comparing representative works in the GNN space, as well as deep non-negative matrix factorisation and sparse filtering-based methods but do not empirically evaluate the methods discussed. Chunaev [16] provides an overview of community detection methods but omits mention of GNN methods. Tu et al. [17] provide an overview of network representation learning for community detection and propose a novel method to incorporate node connectivity and features, but do not compare GNN methods in the study. Ezugwu et al. [18] survey clustering algorithms, providing an in-depth discussion of all applications of clustering algorithms, but do not includes an investigation into methods that perform clustering on graphs. There are many traditional community detection methods that are not based on neural networks. Louvain [19] is a classic community detection algorithm that uses two phases. In the modularity optimisation phase, nodes are randomly ordered then added and removed from communities until there is no significant change in modularity; nodes with the same community are collected and represented as a single node in community aggregation. Leiden [20] is an extension of Louvain, designed to fix the tendency to discover weakly connected communities. Leiden has some subtle differences, which include the addition of a refinement of partitions phase in between the modularity optimisation and community aggregation phases of Louvain. Rather than the greedy merging of Louvain based on the largest increase in modularity, the chance of merging increases in proportion to the modularity. Also, Leiden uses a ‘fast local move procedure’, whereby it only visits the nodes whose neighbourhoods have been changed in the last iteration of the modularity optimisation. Label Propagation [21] uses network structure to find communities of densely connected nodes. This works by initialising every node with a unique label, and at every iteration, each node is reassigned the label that the majority of its neighbours have. These traditional methods are transductive [22], as they do not learn a function that can be applied to unseen data, whereas in this work, we only consider inductive GNN methods which

produce a model that can predict community assignments for data that it has not been trained on.

Various frameworks exist for evaluating performance, and the evaluation procedure employed significantly influences the performance of all algorithms [23]. Under consistent conditions, it has been demonstrated that simple models can exhibit improved performance with a thorough exploration of the hyperparameter space [24]. This improvement may be attributed to the impact of random initialisations on performance [11]. Importantly, relying on results from papers without conducting the same hyperparameter optimisation across all models introduces inconsistency and yields a misleading benchmark. The biased selection of random seeds, which can skew performance, is considered unfair. Furthermore, not training over the same number of epochs or neglecting model selection based on validation set results leads to unfair comparisons, potentially resulting in inaccurate conclusions about the effectiveness of models. Previous work in this space by Leeney and McConville [7] proposed a framework for consistent community detection with GNNs and quantifying the randomness in these investigations. In this work, we expand upon this metric of randomness by considering the effect of ties in performance. We show the effect of ties on ranking randomness and propose two new metrics that improve upon the previous work, evaluating how sensitive each is to the scale of the investigation.

## 2. Methodology

This section details the procedure for evaluation; the problem that is aimed to solve; the hyperparameter optimisation and the resources allocated to this investigation; the algorithms that are being tested; the metrics of performance and datasets used.

The current method of evaluating algorithms suffers from various shortcomings that impede the fairness and reliability of model comparisons. Establishing a consistent framework provides a transparent and objective basis for comparing models. A standardised benchmark practice contributes to transparency by thoroughly documenting the factors influencing performance, encouraging researchers to engage in fair comparisons. To leverage results from previous research, it is essential to follow the exact evaluation procedure, saving time and effort for practitioners. Establishing consistent practices is crucial, as there is currently no reason for confidence in performance claims without a trustworthy evaluation, fostering a deeper understanding and facilitating progress in the field. For this reason, we follow the procedure established by Leeney and McConville [7].

None of the evaluated algorithms are deterministic, as each relies on randomness for initializing the network. Thus, the consistency of a framework can be assessed by considering the amount to which performance rankings change when different randomness is introduced across all tests within the framework. Tests refer to the metric's performance on a specific dataset, and ranking indicates the algorithm's placement relative to others. In this context, different randomness means each distinct random seed used evaluating the algorithms. To evaluate the consistency of results obtained by this framework, we compare the existing coefficient of randomness with two new metrics. We investigate the effect of performance ties on these metrics. In the existing metrics, ties are dealt with by awarding each algorithm the lowest rank between those that share a rank. In this work, this is compared with the scenario of awarding the mean rank of those that are tied. Ties are likely to occur under certain metrics such as conductance, where the algorithm scores the optimal value of 0. In addition, ties will occur where algorithms run out of memory in computation. With this setup, the improved version of Kendall's W coefficient of concordance [25] that can account for ties is used to assess the consistency of rankings. In addition, we use the Wasserstein distance to create another metric for quantifying the difference in rankings due to random seeds. From Leeney and McConville [7], the W randomness coefficient is calculated using the number of algorithms  $a$ , and random seeds  $n$ , along with tests of performance that create rankings of algorithms, as defined by Equation (1):

$$W = 1 - \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{12S}{n^2(a^3 - a)}. \quad (1)$$

The sum of squared deviations  $S$  is calculated for all algorithms, and calculated using the deviation from mean rank due for each random seed. This is averaged over all metrics and datasets that make up the suite of tests  $\mathcal{T}$ . Using the one minus means that if the  $W$  is high, then randomness has affected the rankings, whereas a consistent ranking procedure results in a lower number. By detailing the consistency of a framework across the randomness evaluated, the robustness of the framework can be maintained, allowing researchers to trust results and maintain credibility of their publications. However, this metric does not account for ties in performance and deals with this by assigning the lowest rank from all the algorithms that tie. Instead, an improvement to this is where, when there are ties, each is given the average of the ranks that would have been given had no ties occurred. Where there are a large number of ties, this reduces the value of  $W$  and allows us to compute the correction equation as

$$W_t = 1 - \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{12 \sum_{i=1}^a (R_i^2) - 3n^2a(a+1)^2}{n^2a(a^2-1) - n \sum_{j=1}^n (\sum_{i=1}^{g_j} (t_i^3 - t_i))}, \quad (2)$$

where  $R_i$  is the sum of the ranks for algorithm  $i$ ,  $g_j$  is the number of groups of ties in the rankings under seed  $j$ , and  $t_i$  is the number of ties in that group.

The other metric that we propose is to calculate the overlap in ranking distributions. This is normalised by the maximum difference that would occur when there is no uncertainty in rank due to randomness as there would be no overlap between the ranking distributions. Formally, we calculate the normalised Wasserstein distance [26,27] between rank distributions over each of the test scenarios. Therefore, given the probability distribution of the rank of an algorithm  $j$  as  $f_j(r)$  over the discrete ranking space  $R$ , then the cumulative distribution is denoted as  $F_j(r)$ , and therefore, the Wasserstein distance between two rank distributions is given by Equation (3):

$$W1(i, j) = \int_{R_a} |F_i(r), F_j(r)| dr. \quad (3)$$

This leads us to the definition of the  $W_w$  Wasserstein randomness given by Equation (4),

$$W_w = 1 - \frac{1}{|\mathcal{T}|} \sum_{t \in \mathcal{T}} \frac{\sum_{i=1}^a \sum_{j=1}^{i-1} W1(i, j)}{\sum_{v=1}^a \frac{v(v-1)}{2}}. \quad (4)$$

We also need to compare the effectiveness of these coefficients in assessing the randomness present in the investigations. To do this, we sample a number of tests from the 44 different coefficients ten times for each number of potential tests. This allows us to see how the different coefficients converge to the true coefficient value found by computing the metric over all tests.

To assess the quality of the results, we compare whether the performance is better under the hyperparameters or the default reported by the original implementations. The different parameter sets are given a rank by comparing the performance on every test. This is then averaged across every test, to give the Framework Comparison Rank (FCR) [7]. Demonstrating that failing to optimise hyperparameters properly can result in sub-optimal performance means that models that could have performed better with proper tuning may appear inferior. This affects decision making and potentially leading to the adoption of sub-optimal solutions. In the real world, this can have costly and damaging consequences, and is especially critical in domains where model predictions impact decisions, such as healthcare, finance, and autonomous vehicles.

## 2.1. Problem Definition

The problem definition of community detection on attributed graphs is defined as follows. The graph, where  $N$  is the number of nodes in 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  $d$  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. The only information typically given to the algorithms at training time is the number of clusters  $k$  to partition the graph into. Hard clustering is assumed, where each community detection algorithm must assign each node a single community to which it belongs, such that  $P \in \mathbb{R}^N$ , and we evaluate the clusters associated with each node using the labels given with each dataset, such that  $L \in \mathbb{R}^N$ . Metrics that compare to the ground truth labels are allowed to be used for early stopping and for hyperparameter optimisation.

## 2.2. Hyperparameter Optimisation Procedure

There are sweet spots of architecture combinations that are best for each dataset [28] and the effects of not selecting hyperparameters (HPs) have been well-documented. 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 [29]. Thus, a HPO is performed under feasible constraints in order to validate the hypothesis that HPO affects the comparison of methods. It has been shown that grid search is not suited for searching for HPs on a new dataset and that Bayesian approaches perform better than random [28]. There are a variety of Bayesian methods that can be used for hyperparameter selection. One such is the Tree Parzen-Estimator (TPE) [30] that can retain the conditionality of variables [29] and has been shown to be a good estimator given limited resources [31]. The multi-objective version of the TPE [32] is used to explore the multiple metrics of performance investigated. Given a limited budget, the TPE is optimal, as it allows the efficient exploration of parameters (Table 1).

**Table 1.** Resources are allocated an investigation, and those detailed are shared across all investigations. Algorithms that are designed to benefit from a small number of HPs should perform better, as they can search more of the space within the given budget. All models are trained with 1 × 2080 Ti GPU on a server with 12 GB of RAM, and a 16core Xeon CPU.

Resource	Associated Allocation
Optimiser	Adam
Learning Rate	{0.05, 0.01, 0.005, 0.001, 0.0005, 0.0001}
Weight Decay	{0.05, 0.005, 0.0005, 0.0}
Max Epochs	5000
Patience	{25, 100, 500, 1000}
Max Hyperparameter Trials	300
Seeds	{42, 24, 976, 12345, 98765, 7, 856, 90, 672, 785}
Training/Validation Split	0.8
Train/Testing Split	0.8

In this framework, a modification to the nested cross-validation procedure is used to match reasonable computational budgets, which is to optimise hyperparameters on the first seed tested on and use those hyperparameters on the other seeds. Additionally, it is beneficial to establish a common resource allocation such as number of epochs allowed for in training or number of hyperparameter trials. Ensuring the same resources are used in all investigations means that the relatively underfunded researchers can participate in the field, democratising the access to contribution. Conversely, this also means that highly funded researchers cannot bias their results by exploiting the resources they have available.

## 2.3. Suite of Tests

A test of an algorithm in this framework is the performance under a metric on a dataset. Algorithms are ranked on each test on every random seed used. For evaluation

purposes, some metrics require the ground truth, and others do not, although regardless, this knowledge is not used during the training itself. Macro F1-score (F1) is calculated to ensure that evaluations are not biased by the label distribution, as communities' sizes are often imbalanced. Normalised mutual information (NMI) is also used, which is the amount of information that can be extracted from one distribution with respect to a second.

For unsupervised metrics, modularity and conductance are selected. Modularity quantifies the deviation of the clustering from what would be observed in expectation under a random graph. Conductance is the proportion of total edge volume that points outside the cluster. These two metrics are unsupervised, as they are calculated using the predicted cluster label and the adjacency matrix, without using any ground truth. Many metrics are used in the framework, as they align with specific objectives and ensure that evaluations reflect a clear and understandable assessment of performance.

Generalisation of performance on one dataset can often not be statistically valid and lead to overfitting on a particular benchmark [33]; hence, multiple are used in this investigation. To fairly compare different GNN architectures, a range of graph topologies are used to fairly represent potential applications. Each dataset can be summarised by commonly used graph statistics: the average clustering coefficient [34] and closeness centrality [35]. The former is the proportion of all the connections that exist in a node's neighbourhood compared to a fully connected neighbourhood, averaged across all nodes. The latter is the reciprocal of the mean shortest path distance from all other nodes in the graph. All datasets are publicly available and have been used previously in GNN research [36].

Using many datasets for evaluation means that dataset bias is mitigated, which means that the framework is better at assessing the generalisation capability of models to different datasets. These datasets are detailed in Table 2; the following is a brief summary. Cora [37], CiteSeer [38], and DBLP [39] are graphs of academic publications from various sources with the features coming from words in publications and connectivity from citations. AMAC and AMAP are extracted from the Amazon co-purchase graph [40]. Texas, Wisc, and Cornell are extracted from web pages from computer science departments of various universities [41]. UAT, EAT, and BAT contain airport activity data collected from the National Civil Aviation Agency, Statistical Office of the European Union and Bureau of Transportation Statistics [36].

**Table 2.** The datasets and associated statistics.

Datasets	Nodes	Edges	Features	Classes	Average Clustering Coefficient	Mean Closeness Centrality
AMAC [40]	13752	160124	767	10	0.157	0.264
AMAP [40]	7650	238163	745	8	0.404	0.242
BAT [36]	131	2077	81	4	0.636	0.469
CiteSeer [38]	3327	9104	3703	6	0.141	0.045
Cora [37]	2708	10556	1433	7	0.241	0.137
DBLP [39]	4057	7056	334	4	0.177	0.026
EAT [36]	399	11988	203	4	0.539	0.441
UAT [36]	1190	27198	239	4	0.501	0.332
Texas [41]	183	325	1703	5	0.198	0.344
Wisc [41]	251	515	1703	5	0.208	0.32
Cornell [41]	183	298	1703	5	0.167	0.326

#### 2.4. Models

We consider a representative suite of GNNs, selected based on factors such as code availability and re-implementation time. In addition to explicit community detection algorithms, we also consider those that can learn an unsupervised representation of data, as there is previous research that applies vector-based clustering algorithms to the representations [42]. The following are GNN architectures that learn representations of attributed graphs without a comparison to any labels during the training process. All these methods have hyperparameters which control trade-offs in optimisation.

Deep Attentional Embedded Graph Clustering (DAEGC) uses a k-means target to self-supervise the clustering module to iteratively refine the clustering of node embeddings [43]. Deep Modularity Networks (DMoNs) use GCNs to maximise a modularity-based clustering objective to optimise cluster assignments by a spectral relaxation of the problem [44]. Neighbourhood Contrast Framework for Attributed Graph Clustering (CAGC) [45] is designed for attributed graph clustering with contrastive self-expression loss that selects positive/negative pairs from the data and contrasts representations with its k-nearest neighbours. Deep Graph Infomax (DGI) maximises mutual information between patch representations of sub-graphs and the corresponding high-level summaries [46]. GRaph Contrastive rEpresentation learning (GRACE) generates a corrupted view of the graph by removing edges and learns node representations by maximising agreement across two views [47]. 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 neighbours and a general graph diffusion [48]. Bootstrapped Graph Latents (BGRL) [49] uses a self-supervised bootstrap procedure by maintaining two graph encoders; the online one learns to predict the representations of the target encoder, which in itself is updated by an exponential moving average of the online encoder. SelfGNN [50] also uses this principal but uses augmentations of the feature space to train the network. Towards Unsupervised Deep Graph Structure Learning (SUBLIME) [51] is an encoder with the bootstrapping principle applied over the feature space as well as a contrastive scheme between the nearest neighbours. Variational Graph AutoEncoder Reconstruction (VGAER) [52] reconstructs a modularity distribution using a cross-entropy-based decoder from the encoding of a VGAE [53].

### 3. Evaluation and Discussion

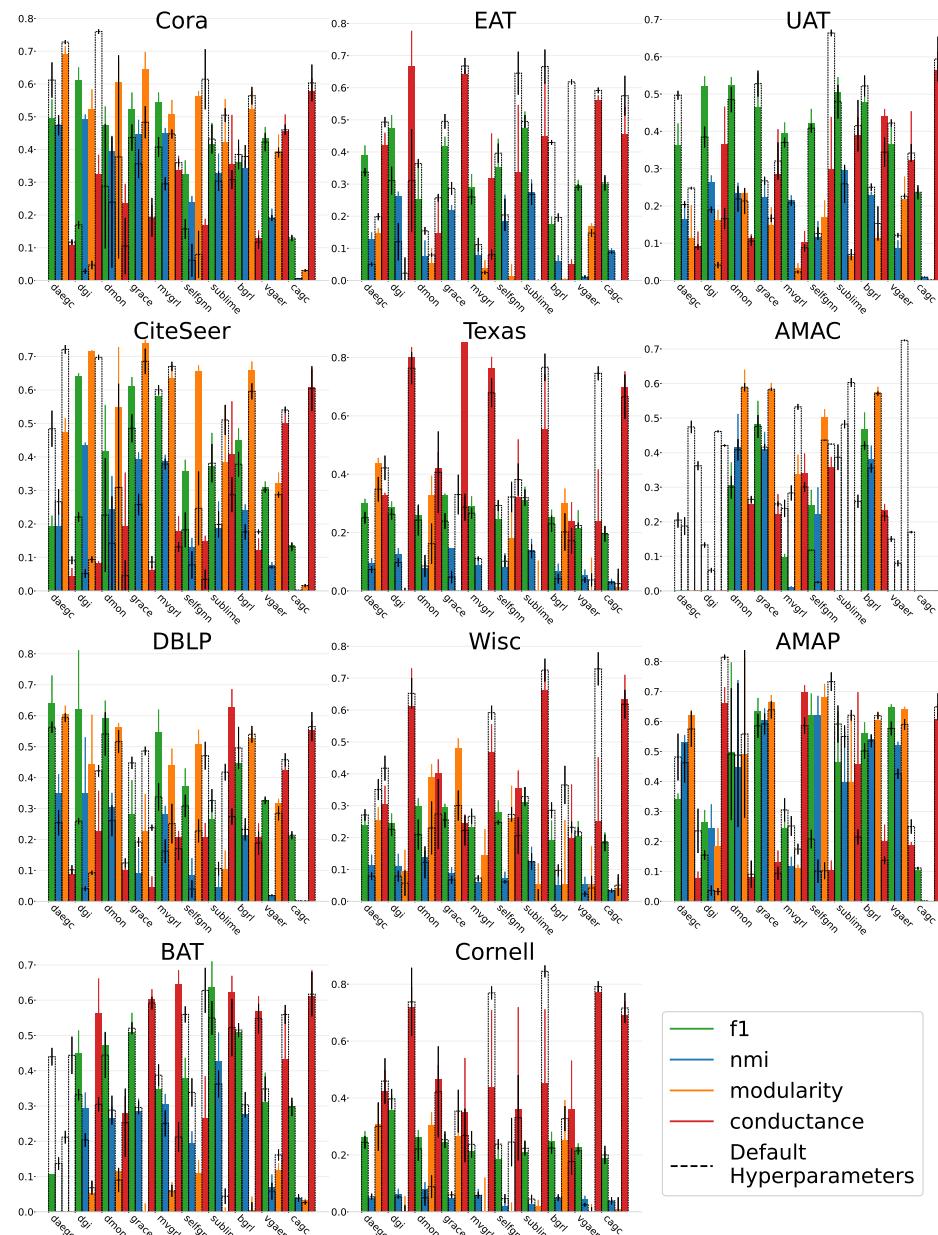
The Framework Comparison Rank is the average rank when comparing performance of the parameters found through hyperparameter optimisation versus the default values. From Table 3, it can be seen that Framework Comparison Rank indicates that the hyperparameters that are optimised on average perform better. This validates the hypothesis that the hyperparameter optimisation significantly impacts the evaluation of GNN-based approaches to community detection.

**Table 3.** Here we show the quantification of intra-framework consistency using the  $W$  randomness coefficient,  $W$  Randomness with Mean Ties, Tied  $W$  Randomness,  $W$  Wasserstein Randomness, and inter-framework disparity using the Framework Comparison Rank. Low values for the Framework Comparison Rank and all  $W$  randomness coefficients are preferred.

Results	Default	HPO
Framework Comparison Rank	1.829	1.171
$W$ Randomness Coefficient	0.476	0.489
N Ties	10	200
$W$ Randomness w/ Mean Ties	0.150	0.245
Tied $W_t$ Randomness	0.150	0.229
$W_w$ Wasserstein Randomness	0.072	0.127

The results of the hyperparameter optimisation and the default parameters are visualised in Figure 1. From this, we can see the difference in performance under both sets of hyperparameters. On some datasets, the default parameters work better than those optimised, which means that under the reasonable budget assumed in this framework, they are not reproducible. This adds strength to the claim that using the default parameters is not credible. Without validation that these parameters can be recovered, results cannot be trusted and mistakes are harder to identify. On the other side of this, often the hyperparameter optimisation performs better than the default values. Algorithms were published knowing these parameters can be tuned to better performance. Using reasonable resources, the performance can be increased, which means that without the optimisation procedure,

inaccurate or misleading comparisons are propagated. Reproducible findings are the solid foundation that allows us to build on previous work and are a necessity for scientific validity.

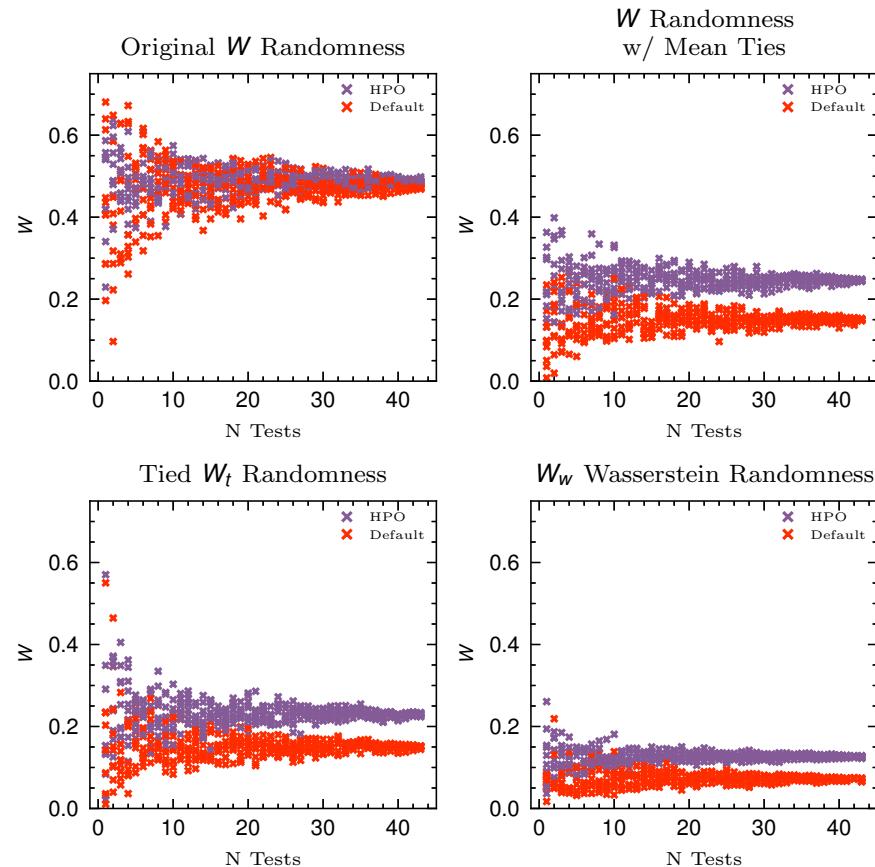


**Figure 1.** The average performance and standard deviation of each metric averaged over every seed tested on for all methods on all datasets. The hyperparameter investigation under our framework is shown in colour compared with the default hyperparameters in dashed boxes. The lower the value for conductance, the better. Out-of-memory occurrences happened on the AMAC dataset with the following algorithms during HPO: DAEGC, SUBLIME, DGI, CAGC, VGAER, and CAGC under the default HPs.

The  $W$  randomness coefficient quantifies the consistency of rankings over the different random seeds tested on, averaged over the suite of tests in the framework. With less deviation of prediction under the presence of randomness, an evaluation finds a more confident assessment of the best algorithm. A marginally higher  $W$  value using the optimised hyperparameters indicates that the default parameters are more consistent over randomness. There is little difference in the coefficients for the original  $W$  randomness coefficient, which is potentially due to the fact that the default parameters have been evaluated with a consistent approach to model selection and constant resource allocation to training time. However,

when we account for the number of tied ranks, we find that the HPO has more tied ranks over the investigation, due to out-of-memory occurrences or optimal conductance values. This may be because the hyperparameter optimisation leads to better results, reducing the difference between algorithms. This is supported by the  $W$  randomness coefficients that account for ties, showing that the HPO is less consistent across randomness. By assessing the  $W$  randomness coefficient, we can reduce the impact of biased evaluation procedures. However, it is important to explain the reasons behind why randomness might be affecting the results. The higher  $W$  for the hyperparameter analysis might be because algorithms increase performance under the HPO. Therefore, when all methods are fairly evaluated under a HPO, the spread of rankings may overlap more, which is not explicitly negative. With this coefficient, researchers can quantify how reliable their results are, and therefore the usability in real-world applications. This sets the baseline for consistency in evaluation procedures and allows better understanding of relative method performance.

In Figure 2, we see the convergence of each metric as the number of tests increases. There is significant overlap between the default and HPO investigation. When we use the original  $W$  randomness coefficient but change how tied ranks are dealt with, by taking the mean of the ties, we can see that the two distributions overlap less. This demonstrates that how ties are dealt with does change quantification of randomness. Comparing this to the  $W_t$  randomness coefficient, we can see that this metric does not affect the measurement of randomness at the scale that is carried out herein. However, when comparing to the  $W_w$ , it is clear that this coefficient has significantly less spread for a low number of tests and converges quicker to the true value, as measured by the full breadth of the investigation. Therefore, this metric can be used to provide a more accurate assessment of the amount of randomness in the investigation and the robustness of the methods evaluated.



**Figure 2.** Here, the different metrics of quantifying randomness are compared against samples of the original testing space. The randomness in rankings over samples of experiments using the default hyperparameters is compared against the HPO results. The distinction between the original  $W$  randomness coefficient proposed by Leeney and McConvile [7] and the other metrics of randomness is that ties are settled by taking the mean of the ranks rather than assigning the lowest rank to all algorithms.

Given different computational resources, performance rankings will vary. Future iterations of the framework should experiment with the number of trials and impact of over-reliance on specific seeds or extending the hyperparameter options to a continuous distribution. Additionally, finding the best general algorithm will have to include a wide range of different topologies or sizes of graphs that are not looked at, and we do not explore other feature space landscapes or class distributions.

#### 4. Conclusions

In this work, we demonstrate flaws with how GNN-based community detection methods are currently evaluated, leading to potentially misleading and confusing conclusions. To address this, a framework was compared for providing a more consistent and fair evaluation for GNN community detection. We provide further insight that consistent HPO is key in this task by quantifying the difference in performance from HPO to reported values. Finally, different metrics of assessing the consistency of rankings under the presence of randomness are compared. It is found that the  $W$  Wasserstein randomness coefficient is the most robust to samples of the full test investigation.

**Author Contributions:** Conceptualisation, W.L. and R.M.; methodology, W.L.; software, W.L.; validation, W.L. and R.M.; formal analysis, W.L.; investigation, W.L.; resources, W.L.; data curation, W.L.; writing—original draft preparation, W.L.; writing—review and editing, W.L. and R.M.; visualisation, W.L.; supervision, R.M.; project administration, R.M.; funding acquisition, R.M. All authors have read and agreed to the published version of the manuscript.

**Funding:** This work was funded by the EPSRC.

**Data Availability Statement:** Data are publicly available at <https://github.com/yueliu1999/Awesome-Deep-Graph-Clustering>, (accesssed on 13 December 2023).

**Conflicts of Interest:** The authors declare no conflicts of interest.

#### Abbreviations

The following abbreviations are used in this manuscript:

HPO	Hyperparameter Optimisation
HPs	Hyperparameters
F1	Macro F1-score
NMI	Normalised Mutual Information
GNN	Graph Neural Network
FCR	Framework Comparison Rank
DAEGC	Deep Attentional Embedded Graph Clustering
DMoN	Deep Modularity Network
CAGC	Neighbourhood Contrast Framework for Attributed Graph Clustering
DGI	Deep Graph Infomax
GRACE	GRApH Contrastive rEpresentation learning
MVGRL	Contrastive Multi-View Representation Learning on Graphs
BGRL	Bootstrapped Graph Latents
SUBLIME	Towards Unsupervised Deep Graph Structure Learning
VGAER	Variational Graph AutoEncoder Reconstruction

#### References

1. Kipf, T.N.; Welling, M. Semi-supervised classification with graph convolutional networks. *arXiv* **2016**, arXiv:1609.02907.
2. Schaeffer, S.E. Graph clustering. *Comput. Sci. Rev.* **2007**, *1*, 27–64. [CrossRef]
3. Monti, F.; Frasca, F.; Eynard, D.; Mannion, D.; Bronstein, M.M. Fake news detection on social media using geometric deep learning. *arXiv* **2019**, arXiv:1902.06673.
4. Cabreros, I.; Abbe, E.; Tsirigos, A. Detecting community structures in hi-c genomic data. In Proceedings of the 2016 Annual Conference on Information Science and Systems (CISS), Princeton, NJ, USA, 16–18 March 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 584–589.

5. Yang, J.; McAuley, J.; Leskovec, J. Community detection in networks with node attributes. In Proceedings of the 2013 IEEE 13th International Conference on Data Mining, Dallas, TX, USA, 7–10 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 1151–1156.
6. Hu, W.; Fey, M.; Ren, H.; Nakata, M.; Dong, Y.; Leskovec, J. Ogb-lsc: A large-scale challenge for machine learning on graphs. *arXiv* **2021**, arXiv:2103.09430.
7. Leeney, W.; McConville, R. Uncertainty in GNN Learning Evaluations: The Importance of a Consistent Benchmark for Community Detection. In Proceedings of the Twelfth International Conference on Complex Networks & Their Applications, Menton Riviera, France, 28–30 November 2023; Springer: Berlin/Heidelberg, Germany, 2023.
8. Pineau, J.; Vincent-Lamarre, P.; Sinha, K.; Larivière, V.; Beygelzimer, A.; d'Alché Buc, F.; Fox, E.; Larochelle, H. Improving reproducibility in machine learning research (a report from the neurips 2019 reproducibility program). *J. Mach. Learn. Res.* **2021**, *22*, 7459–7478.
9. Dwivedi, V.P.; Joshi, C.K.; Laurent, T.; Bengio, Y.; Bresson, X. Benchmarking graph neural networks. *arXiv* **2020**, arXiv:2003.00982.
10. Morris, C.; Kriege, N.M.; Bause, F.; Kersting, K.; Mutzel, P.; Neumann, M. Tudataset: A collection of benchmark datasets for learning with graphs. *arXiv* **2020**, arXiv:2007.08663.
11. Errica, F.; Podda, M.; Bacciu, D.; Micheli, A. A fair comparison of graph neural networks for graph classification. *arXiv* **2019**, arXiv:1912.09893.
12. Palowitch, J.; Tsitsulin, A.; Mayer, B.; Perozzi, B. GraphWorld: Fake Graphs Bring Real Insights for GNNs. *arXiv* **2022**, arXiv:2203.00112.
13. Liu, F.; Xue, S.; Wu, J.; Zhou, C.; Hu, W.; Paris, C.; Nepal, S.; Yang, J.; Yu, P.S. Deep learning for community detection: progress, challenges and opportunities. *arXiv* **2020**, arXiv:2005.08225.
14. Jin, D.; Yu, Z.; Jiao, P.; Pan, S.; He, D.; Wu, J.; Yu, P.; Zhang, W. A survey of community detection approaches: From statistical modeling to deep learning. *IEEE Trans. Knowl. Data Eng.* **2021**, *35*, 1149–1170. [[CrossRef](#)]
15. Su, X.; Xue, S.; Liu, F.; Wu, J.; Yang, J.; Zhou, C.; Hu, W.; Paris, C.; Nepal, S.; Jin, D.; et al. A comprehensive survey on community detection with deep learning. *IEEE Trans. Neural Netw. Learn. Syst.* **2022**, *early access*. [[CrossRef](#)] [[PubMed](#)]
16. Chunaev, P. Community detection in node-attributed social networks: A survey. *Comput. Sci. Rev.* **2020**, *37*, 100286. [[CrossRef](#)]
17. Tu, C.; Zeng, X.; Wang, H.; Zhang, Z.; Liu, Z.; Sun, M.; Zhang, B.; Lin, L. A unified framework for community detection and network representation learning. *IEEE Trans. Knowl. Data Eng.* **2018**, *31*, 1051–1065. [[CrossRef](#)]
18. Ezugwu, A.E.; Ikofun, A.M.; Oyelade, O.O.; Abualigah, L.; Agushaka, J.O.; Eke, C.I.; Akinyelu, A.A. A comprehensive survey of clustering algorithms: State-of-the-art machine learning applications, taxonomy, challenges, and future research prospects. *Eng. Appl. Artif. Intell.* **2022**, *110*, 104743. [[CrossRef](#)]
19. Blondel, V.D.; Guillaume, J.L.; Lambiotte, R.; Lefebvre, E. Fast unfolding of communities in large networks. *J. Stat. Mech. Theory Exp.* **2008**, *2008*, P10008. [[CrossRef](#)]
20. Traag, V.A.; Waltman, L.; Van Eck, N.J. From Louvain to Leiden: guaranteeing well-connected communities. *Sci. Rep.* **2019**, *9*, 5233. [[CrossRef](#)]
21. Raghavan, U.N.; Albert, R.; Kumara, S. Near linear time algorithm to detect community structures in large-scale networks. *Phys. Rev. E* **2007**, *76*, 036106. [[CrossRef](#)]
22. Vapnik, V. *Statistical Learning Theory*; Wiley-Interscience: New York, NY, USA, 1998.
23. Zöller, M.A.; Huber, M.F. Benchmark and survey of automated machine learning frameworks. *J. Artif. Intell. Res.* **2021**, *70*, 409–472. [[CrossRef](#)]
24. Shchur, O.; Mumme, M.; Bojchevski, A.; Günnemann, S. Pitfalls of graph neural network evaluation. *arXiv* **2018**, arXiv:1811.05868.
25. Field, A.P. Kendall's coefficient of concordance. *Encycl. Stat. Behav. Sci.* **2005**, *2*, 1010–1011.
26. Vallender, S. Calculation of the Wasserstein distance between probability distributions on the line. *Theory Probab. Its Appl.* **1974**, *18*, 784–786. [[CrossRef](#)]
27. Shen, J.; Qu, Y.; Zhang, W.; Yu, Y. Wasserstein distance guided representation learning for domain adaptation. In Proceedings of the AAAI Conference on Artificial Intelligence, New Orleans, LA, USA, 2–7 February 2018; Volume 32.
28. Bergstra, J.; Bengio, Y. Random search for hyper-parameter optimization. *J. Mach. Learn. Res.* **2012**, *13*, 281–305.
29. Yang, L.; Shami, A. On hyperparameter optimization of machine learning algorithms: Theory and practice. *Neurocomputing* **2020**, *415*, 295–316. [[CrossRef](#)]
30. Bergstra, J.; Yamins, D.; Cox, D. Making a science of model search: Hyperparameter optimization in hundreds of dimensions for vision architectures. In Proceedings of the International Conference on Machine Learning, PMLR, Atlanta, GA, USA, 16–21 June 2013; pp. 115–123.
31. Yuan, Y.; Wang, W.; Pang, W. A systematic comparison study on hyperparameter optimisation of graph neural networks for molecular property prediction. In Proceedings of the Genetic and Evolutionary Computation Conference, Lille, France, 10–14 July 2021; pp. 386–394.
32. Ozaki, Y.; Tanigaki, Y.; Watanabe, S.; Onishi, M. Multiobjective tree-structured parzen estimator for computationally expensive optimization problems. In Proceedings of the 2020 Genetic and Evolutionary Computation Conference, Cancun, Mexico, 8–12 July 2020; pp. 533–541.
33. Salzberg, S.L. On comparing classifiers: Pitfalls to avoid and a recommended approach. *Data Min. Knowl. Discov.* **1997**, *1*, 317–328. [[CrossRef](#)]
34. Watts, D.J.; Strogatz, S.H. Collective dynamics of ‘small-world’ networks. *Nature* **1998**, *393*, 440–442. [[CrossRef](#)] [[PubMed](#)]
35. Wasserman, S.; Faust, K. *Social Network Analysis: Methods and Applications*; Cambridge University Press: Cambridge, UK, 1994.

36. Liu, Y.; Xia, J.; Zhou, S.; Wang, S.; Guo, X.; Yang, X.; Liang, K.; Tu, W.; Li, Z.S.; Liu, X. A Survey of Deep Graph Clustering: Taxonomy, Challenge, and Application. *arXiv* **2022**, arXiv:2211.12875.
37. McCallum, A.K.; Nigam, K.; Rennie, J.; Seymore, K. Automating the construction of internet portals with machine learning. *Inf. Retr.* **2000**, *3*, 127–163. [[CrossRef](#)]
38. Giles, C.L.; Bollacker, K.D.; Lawrence, S. CiteSeer: An automatic citation indexing system. In Proceedings of the Third ACM Conference on Digital Libraries, Pittsburgh, PA, USA, 23–26 June 1998; pp. 89–98.
39. Tang, J.; Zhang, J.; Yao, L.; Li, J.; Zhang, L.; Su, Z. Arnetminer: Extraction and mining of academic social networks. In Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, NV, USA, 24–27 August 2008; pp. 990–998.
40. He, R.; McAuley, J. 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, Montreal, QC, Canada, 11–15 April 2016; pp. 507–517.
41. Craven, M.; McCallum, A.; PiPasquo, D.; Mitchell, T.; Freitag, D. *Learning to Extract Symbolic Knowledge from the World Wide Web*; Technical Report; Carnegie Mellon School of Computer Science: Pittsburgh, PA, USA, 1998.
42. McConville, R.; Santos-Rodriguez, R.; Piechocki, R.J.; Craddock, I. N2d:(Not too) deep clustering via clustering the local manifold of an autoencoded embedding. In Proceedings of the 2020 25th International Conference on Pattern Recognition (ICPR), Milan, Italy, 10–15 January 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 5145–5152.
43. Wang, C.; Pan, S.; Hu, R.; Long, G.; Jiang, J.; Zhang, C. Attributed graph clustering: A deep attentional embedding approach. *arXiv* **2019**, arXiv:1906.06532.
44. Tsitsulin, A.; Palowitch, J.; Perozzi, B.; Müller, E. Graph clustering with graph neural networks. *J. Mach. Learn. Res.* **2023**, *24*, 1–21.
45. Wang, T.; Yang, G.; He, Q.; Zhang, Z.; Wu, J. NCAGC: A Neighborhood Contrast Framework for Attributed Graph Clustering. *arXiv* **2022**, arXiv:2206.07897.
46. Velickovic, P.; Fedus, W.; Hamilton, W.L.; Liò, P.; Bengio, Y.; Hjelm, R.D. Deep Graph Infomax. *ICLR (Poster)* **2019**, *2*, 4.
47. Zhu, Y.; Xu, Y.; Yu, F.; Liu, Q.; Wu, S.; Wang, L. Deep graph contrastive representation learning. *arXiv* **2020**, arXiv:2006.04131.
48. Hassani, K.; Khasahmadi, A.H. Contrastive multi-view representation learning on graphs. In Proceedings of the International Conference on Machine Learning, PMLR, Virtual, 13–18 July 2020; pp. 4116–4126.
49. Thakoor, S.; Tallec, C.; Azar, M.G.; Munos, R.; Veličković, P.; Valko, M. Bootstrapped representation learning on graphs. In Proceedings of the ICLR 2021 Workshop on Geometrical and Topological Representation Learning, 2021.
50. Kefato, Z.T.; Girdzijauskas, S. Self-supervised graph neural networks without explicit negative sampling. *arXiv* **2021**, arXiv:2103.14958.
51. Liu, Y.; Zheng, Y.; Zhang, D.; Chen, H.; Peng, H.; Pan, S. Towards unsupervised deep graph structure learning. In Proceedings of the ACM Web Conference 2022, Lyon, France, 25–29 April 2022; pp. 1392–1403.
52. Qiu, C.; Huang, Z.; Xu, W.; Li, H. VGAER: Graph Neural Network Reconstruction based Community Detection. In Proceedings of the AAAI: DLG-AAAI'22, Virtual, 28 February–1 March 2022.
53. Kipf, T.N.; Welling, M. Variational graph auto-encoders. *arXiv* **2016**, arXiv:1611.07308.

**Disclaimer/Publisher’s Note:** The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.