

Graph Clustering with Graph Neural Networks

Anton Tsitsulin *Google Research, New York, NY, USA*

TSITSULIN@GOOGLE.COM

John Palowitch *Google Research, San Francisco, CA, USA*

PALOWITCH@GOOGLE.COM

Bryan Perozzi *Google Research, New York, NY, USA*

BPEROZZI@ACM.ORG

Emmanuel Müller

EMMANUEL.MUELLER@CS.TU-DORTMUND.DE

Technical University of Dortmund, Germany

Editor: Honglak Lee

Abstract

Graph Neural Networks (GNNs) have achieved state-of-the-art results on many graph analysis tasks such as node classification and link prediction. However, important unsupervised problems on graphs, such as graph clustering, have proved more resistant to advances in GNNs. Graph clustering has the same overall goal as node pooling in GNNs—does this mean that GNN pooling methods do a good job at clustering graphs?

Surprisingly, the answer is no—current GNN pooling methods often fail to recover the cluster structure in cases where simple baselines, such as k-means applied on learned representations, work well. We investigate further by carefully designing a set of experiments to study different signal-to-noise scenarios both in graph structure and attribute data. To address these methods’ poor performance in clustering, we introduce Deep Modularity Networks (DMON), an unsupervised pooling method inspired by the modularity measure of clustering quality, and show how it tackles recovery of the challenging clustering structure of real-world graphs. Similarly, on real-world data, we show that DMON produces high quality clusters which correlate strongly with ground truth labels, achieving state-of-the-art results with over **40%** improvement over other pooling methods across different metrics.

Keywords: Graph Clustering, Graph Neural Networks, Stochastic Block Models

1. Introduction

In recent years there has been a surge of research interest in developing varieties of Graph Neural Networks (GNNs)—specialized deep learning architectures for dealing with graph-structured data, such as social networks (Perozzi et al., 2014b), recommender graphs (Ying et al., 2018a), or molecular graphs (Defferrard et al., 2016). GNNs leverage the structure of the data as a computational graph, allowing the information to propagate across the edges of graphs (Scarselli et al., 2008). When many real-world systems are represented as graphs, they exhibit locally inhomogeneous distributions of edges, forming clusters (also called communities or modules)—groups of nodes with high in-group edge density, and relatively low out-group density. Clusters can correspond to interesting phenomena in the underlying graph, for example to education (Traud et al., 2011) or employment (Papacharissi, 2009) in social graphs. GNNs have been shown to benefit from leveraging higher-order structural information that could arise from clusters (Chen et al., 2018; Ying et al., 2018b), for example through pooling or trainable attention over edges (Veličković et al., 2017).

Table 1: Related work in terms of six desirable clustering properties outlined in Section 2.

method	End-to-end	✓ required for clustering						Complexity
		Unsup.	Node pooling	Sparse	Soft assign.	Stable		
Graclus	✗	✓	✓	✓	✗	✓		$\mathcal{O}(dn + m)$
DiffPool	✓	✓	✓	✗	✓	✗		$\mathcal{O}(dn^2)$
AGC	✗	✓	✓	✗	✗	✗		$\mathcal{O}(dn^2k)$
DAEGC	✗	✓	✓	✗	✗	✗		$\mathcal{O}(dn^k)$
SDCN	✗	✓	✓	✓	✗	✗		$\mathcal{O}(d^2n + m)$
NOCD	✓	✓	✓	✗	✓	✓		$\mathcal{O}(dn + m)$
Top-k	✓	✗	✗	✓	✗	✓		$\mathcal{O}(dn + m)$
SAG	✓	✗	✗	✓	✗	✗		$\mathcal{O}(dn + m)$
MinCut	✓	✓	✓	✓	✓	✗		$\mathcal{O}(d^2n + m)$
DMoN	✓	✓	✓	✓	✓	✓		$\mathcal{O}(d^2n + m)$

Interestingly, most existing work on GNNs to leverage higher-order structure does not directly address node partitioning or cluster assignments within the computational graph. Furthermore, most works explore these mechanisms only within a semi-supervised or supervised framework, ignoring the fact that *unsupervised* graph clustering is often an extremely useful end-goal in itself—whether for data exploration (Perozzi and Akoglu, 2018), visualization (Cléménçon et al., 2012; Cui et al., 2008), genomic feature discovery (Cabreros et al., 2016), anomaly detection (Perozzi and Akoglu, 2016), or for many other use-cases discussed e.g. in Fortunato and Hric (2016). Additionally, many of the existing unsupervised structure-aware methods have undesirable properties, such as relying on a multi-step optimization process which does not allow for an end-to-end differentiable objective (Perozzi et al., 2014a).

In this work, we take an *ab initio* approach to the clustering problem in the GNN domain, bridging the gap between traditional graph clustering objectives and deep neural networks. We start by drawing a connection between graph pooling, which was typically studied in the literature as a regularizer for supervised GNN architectures, and fully unsupervised clustering. Specifically, we contribute:

- DMoN, an unsupervised clustering module for GNNs that allows optimization of cluster assignments in an end-to-end differentiable way with strong empirical performance.
- An empirical study of performance on synthetic graphs, illustrating the problems with existing work and how DMoN allows for improved model performance in those regimes.
- Thorough experimental evaluation on real-world data, showing that many pooling methods poorly reflect hierarchical structures and are not able to make use of either graph structure and node attributes nor leverage joint information.

2. Related Work

We build upon a rich line of research on graph neural networks and graph pooling methods.

Graph Neural Networks (GNNs) (Scarselli et al., 2008; Duvenaud et al., 2015; Niepert et al., 2016; Gilmer et al., 2017; Kipf and Welling, 2017) allow end-to-end differentiable losses over data with arbitrary structure. They have been applied to an incredible range of applications, from social networks (Perozzi et al., 2014b), to recommender systems (Ying et al., 2018a), to computational chemistry (Gilmer et al., 2017). While GNNs are flexible

enough to allow for unsupervised losses, most work follows the semi-supervised setting for node classification from (Kipf and Welling, 2017). For an introduction to the vast topic we refer to detailed surveys (Bronstein et al., 2017; Hamilton et al., 2017; Chami et al., 2022).

Unsupervised training of GNNs is commonly done via maximizing mutual information (Belghazi et al., 2018; Hjelm et al., 2019; Tschannen et al., 2020; Song and Ermon, 2020) in a self-supervised fashion. Deep Graph Infomax (DGI) (Veličković et al., 2019) adapted the mutual information-based learning from Deep InfoMax (Hjelm et al., 2019), learning unsupervised representations for nodes in attributed graphs. InfoGraph (Sun et al., 2020) extended the idea to learning representations of whole graphs instead of nodes. A very similar approach was independently proposed by Hu* et al. (2020) in the context of pre-training GNNs for producing graph representations, which was first studied in Navarin et al. (2018).

Graph pooling aims to tackle the hierarchical nature of graphs via iterative coarsening. Early architectures (Defferrard et al., 2016) resorted to fixed axiomatic pooling, with no optimization of clustering while the network learns. DiffPool (Ying et al., 2018b) suggests to include a *learnable* pooling to GNN architecture. To help the convergence, DiffPool includes a link prediction loss to help encapsulate the clustering structure of graphs and an additional entropy loss to penalize soft assignments. Top-k (Gao and Ji, 2019) and SAG pooling (Lee et al., 2019) learn to sparsify the graph (select top-k edges for each node) with learned weights. MinCutPool (Bianchi et al., 2020) pooling studies a differentiable formulation of spectral clustering as a pooling strategy. As we show in Section 4.2, MinCutPool does not optimize its spectral objective, it merely orthogonalizes its features.

We summarize mainstream graph pooling methods in Table 1 in terms of seven desirable properties related to their clustering capabilities:

- **End-to-end training** allows to capture both graph structure and node features. All of AGC (Zhang et al., 2019), DAEGC (Wang et al., 2019a), and SDCN (Bo et al., 2020) use k-means to initialize the model, prohibiting end-to-end learning.
- **Unsupervised** training is a desirable setting for clustering models. Works on *supervised* graph clustering (Yang et al., 2019; Wang et al., 2019b) are outside of our scope.
- **Node aggregation** is crucial for our interpretation of graph pooling in terms of clustering. Both Top-k and SAG pooling only sparsify the graph and do not reduce the nodeset.
- **Sparse.** As graphs in the real-world vary in size and sparsity, methods cannot be limited by an $\mathcal{O}(n^2)$ link prediction objectives, like DiffPool (Ying et al., 2018b), or computing \mathbf{A}^t , like top-k pooling methods (Gao and Ji, 2019; Lee et al., 2019) or AGC and DAEGC. The gradients of NOCD (Shchur and Günnemann, 2019) are quadratic in the number of nodes; however, subsampling is solved to attain desirable subquadratic scalability.
- **Soft assignments** allow for more flexible reasoning about the interactions of clusters.
- **Stable** – the method should be stable in terms of the graph structure. DiffPool, AGC, DAEGC, and SDCN change their performance significantly as the graph becomes more sparse, and MinCutPool fails to converge on power-law graphs.

We additionally analyze the computational complexity of the methods. Generally, non-sparse methods, for instance DiffPool, AGC, and DAEGC can not scale to large graphs, since their complexity is at least quadratic. Some methods, like NOCD, top-k pooling, and SAG subsample the quadratic terms in their objectives/optimizations to attain desirable scalability. We analyze the complexity of DMoN in Section 4.1. DMoN maintains best-in-class scalability without sacrificing any information to subsampling.

Graph embeddings (Perozzi et al., 2014b; Grover and Leskovec, 2016; Tsitsulin et al., 2018) can be thought of as (very restricted) unsupervised GNNs with an identity feature matrix, meaning each node learns its own positional representation (You et al., 2019). The learning process in graph embeddings is often done in a similar way to DGI through noise contrastive estimation (Gutmann and Hyvärinen, 2010). As far as we know, all pooling strategies for learning node embeddings without attributes have been rigid (Chen et al., 2018; Liang et al., 2021; Deng et al., 2020); making learning-based ones an interesting future research direction.

3. Preliminaries

We introduce the necessary background for DMoN, starting with the problem formulation, reviewing common graph clustering objectives and how they can be made differentiable.

A graph $G = (V, E)$ is defined via a set of nodes $V = (v_1, \dots, v_n), |V| = n$ and edges $E \subseteq V \times V, |E| = m$. We denote the $n \times n$ adjacency matrix of G by \mathbf{A} , where $\mathbf{A}_{ij} = 1$ iff $\{v_i, v_j\} \in E$ (otherwise entries of \mathbf{A} are equal to 0). The “degree” of v_i is its number of connections $d_i := \sum_{j=1}^n \mathbf{A}_{ij}$. We are interested in measuring the quality of graph partitioning function $\mathcal{F} : V \mapsto \{1, \dots, k\}$ that splits the set of nodes V into k partitions $V_i = \{v_j : \mathcal{F}(v_j) = i\}$. In contrast to standard graph clustering, we are also provided with node attributes $\mathbf{X} \in \mathbb{R}^{n \times s}$.

3.1 Graph Clustering Quality Functions

As classical clustering objectives are discrete and therefore unsuitable for gradient-based optimization, DMoN and the few prior works depend on spectral approximations. To contextualize and motivate our contributions, we review two families of clustering quality functions amenable to spectral optimization, and review some of their shortcomings.

Cut-based metrics. In his seminal work, Fiedler (1973) suggested that the second eigenvector of a graph Laplacian produces a graph *cut* minimal in terms of the weight of the edges. This plain notion of cut degenerates on real-world graphs, as it does not require partitions to be balanced in terms of size. It is possible to get normalized partitions with the use of ratio cut (Wei and Cheng, 1989), which normalizes the cut by the product of the number of nodes in two partitions, or normalized cut (Shi and Malik, 2000), which uses total edge volume of the partition as normalization.

In real networks, however, there is evidence *against* existence of good cuts (Leskovec et al., 2008) in ground-truth communities. This can be explained by the fact that a single node implicitly participates in many different clusters (Epasto et al., 2017), e.g. a person in a social network is simultaneously connected with family and work friends, forcing the algorithm to merge these communities together.

Recently, MinCutPool (Bianchi et al., 2020) adapted the notion of the normalized cut to use as a regularizer for pooling. While MinCutPool’s objective should, theoretically, be suitable for clustering nodes in graphs, we show that MinCutPool does not optimize its own objective function, using synthetic and real-world experiments.

Modularity. The modularity (Newman, 2006a) objective approaches the same problem from a statistical perspective, incorporating a *null model* to quantify the deviation of the

clustering from what would be observed in expectation under a random graph. In a fully random graph with given degrees, nodes u and v with degrees d_u and d_v are connected with probability $d_u d_v / 2m$. Modularity measures the divergence between the intra-cluster edges from the expected one:

$$\mathcal{Q} = \frac{1}{2m} \sum_{ij} \left[\mathbf{A}_{ij} - \frac{d_i d_j}{2m} \right] \delta(c_i, c_j), \quad (1)$$

where $\delta(c_i, c_j) = 1$ if i and j are in the same cluster and 0 otherwise. Note $\mathcal{Q} \in (-1/2; 1]$ (it is 0 when there is no correlation of clusters with edge density), but it is not necessarily maximized at 1, and is only comparable across graphs with the same degree distribution. Moreover, optimal modularity is positive even for Erdős–Rényi random graphs (McDiarmid and Skerman, 2020), meaning positive modularity does not imply strong clustering structure. While problems with the modularity metric have been identified (Good et al., 2010), it remains one of the most commonly-used and eminently useful graph clustering metrics in scientific literature (Fortunato and Hric, 2016).

3.2 Spectral Modularity Maximization

Maximizing the modularity is proven to be **NP**-hard (Brandes et al., 2006), however, a spectral relaxation of the problem can be solved efficiently (Newman, 2006b). Let $\mathbf{C} \in \{0, 1\}^{n \times k}$ be the cluster assignment matrix and \mathbf{d} be the degree vector. Then, with *modularity matrix* \mathbf{B} defined as $\mathbf{B} = \mathbf{A} - \frac{\mathbf{d}\mathbf{d}^\top}{2m}$, the modularity \mathcal{Q} can be reformulated as:

$$\mathcal{Q} = \frac{1}{2m} \text{Tr}(\mathbf{C}^\top \mathbf{B} \mathbf{C}) \quad (2)$$

Relaxing $\mathbf{C} \in \mathbb{R}^{n \times k}$, the optimal \mathbf{C} maximizing \mathcal{Q} is the top-k eigenvectors of the modularity matrix \mathbf{B} . While \mathbf{B} is dense, iterative eigenvalue solvers can take advantage of the fact that \mathbf{B} is a sum of a sparse \mathbf{A} and rank-one matrix $-\frac{\mathbf{d}\mathbf{d}^\top}{2m}$, meaning that the matrix-vector product $\mathbf{B}\mathbf{x}$ can be computed efficiently as

$$\mathbf{B}\mathbf{x} = \mathbf{A}\mathbf{x} - \frac{\mathbf{d}^\top \mathbf{x} \mathbf{d}}{2m}$$

and optimized efficiently with iterative methods such as power iteration or Lanczos algorithm. One can then obtain clusters by means of spectral bisection (Newman, 2006b) with iterative refinement akin to Kernighan-Lin algorithm (Kernighan and Lin, 1970). However, these formulations operate entirely on the graph structure, and it is non-trivial to adapt them to work with attributed graphs.

3.3 Graph Neural Networks

Graph Neural Networks are a flexible class of models that perform nonlinear feature aggregation with respect to graph structure. For the purposes of this work, we consider transductive GNNs that output a single embedding per node. Graph convolutional networks (GCNs) Kipf and Welling (2017) are simple yet effective Shchur et al. (2018) message-passing networks

that fit our criteria. Let $\mathbf{X}^0 \in \mathbb{R}^{n \times s}$ be the initial node features and $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}} \mathbf{A} \mathbf{D}^{-\frac{1}{2}}$ be the normalized adjacency matrix, the output of t -th layer \mathbf{X}^{t+1} is

$$\mathbf{X}^{t+1} = \text{SeLU}(\tilde{\mathbf{A}} \mathbf{X}^t \mathbf{W} + \mathbf{X} \mathbf{W}_{\text{skip}}) \quad (3)$$

We make two changes to the classic GCN architecture: first, we remove the self-loop creation and instead use an $\mathbf{W}_{\text{skip}} \in \mathbb{R}^{s \times s}$ trainable skip connection, and, second, we replace ReLU nonlinearity with SeLU (Klambauer et al., 2017) for better convergence.

4. Method

In this section, we present DMoN, our method for attributed graph clustering with graph neural networks. Inspired by the modularity quality function and its spectral optimization, we propose a fully differentiable unsupervised clustering objective which optimizes soft cluster assignments using a null model to control for inhomogeneities in the graph. We then discuss the challenge of regularizing cluster assignments, and present collapse regularization that is effective at preventing trivial solutions without compromising optimization of the objective.

4.1 DMoN: Deep Modularity Networks

Our approach to GNN modularity is comprised of (1) the architecture to encode the cluster assignments \mathbf{C} , and (2) the objective function with which to optimize the assignments. We propose to obtain \mathbf{C} via the output of a softmax function, which allows the (soft) cluster assignment to be differentiable. The input to the cluster assignment can be any differentiable message passing function, but here we specifically consider the case where a graph convolutional network is used to obtain soft clusters for each node as follows:

$$\mathbf{C} = \text{softmax}(\text{GCN}(\tilde{\mathbf{A}}, \mathbf{X})), \quad (4)$$

where GCN is a (possibly) multi-layer convolutional network operating on a normalized adjacency matrix $\tilde{\mathbf{A}} = \mathbf{D}^{-\frac{1}{2}}(\mathbf{A})\mathbf{D}^{-\frac{1}{2}}$.

We then propose to optimize this assignment with the following objective, which combines insights from spectral modularity maximization (recall Eq. (2)) with a novel regularization to prevent trivial solutions to the optimization problem:

$$\mathcal{L}_{\text{DMoN}}(\mathbf{C}; \mathbf{A}) = \underbrace{-\frac{1}{2m} \text{Tr}(\mathbf{C}^\top \mathbf{B} \mathbf{C})}_{\text{modularity}} + \underbrace{\frac{\sqrt{k}}{n} \left\| \sum_i \mathbf{C}_i^\top \right\|_F - 1}_{\text{collapse regularization}}, \quad (5)$$

where $\|\cdot\|_F$ is the Frobenius norm. We decompose the computation of $\text{Tr}(\mathbf{C}^\top \mathbf{B} \mathbf{C})$ as a sum of sparse matrix-matrix multiplication and rank-one degree normalization $\text{Tr}(\mathbf{C}^\top \mathbf{A} \mathbf{C} - \mathbf{C}^\top \mathbf{d}^\top \mathbf{d} \mathbf{C})$. This brings the time complexity of computing $\mathcal{L}_{\text{DMoN}}$ down from $\mathcal{O}(n^2)$ to $\mathcal{O}(d^2 n)$ per update, allowing DMoN to efficiently work with sparse graphs.

4.2 Collapse regularization

Without additional constraints on the assignment matrix \mathbf{C} , spectral clustering for both min-cut and modularity objectives has spurious local minima: assigning all nodes to the same

Table 2: Comparison of MinCutPool with using **only** its orthogonality regularization in terms of graph conductance \mathcal{C} , modularity \mathcal{Q} , and NMI with ground-truth labels.

method	Cora			Citeseer			Pubmed			Coauthor CS			Coauthor Phys		
	graph	labels	$\mathcal{C} \downarrow$	graph	labels	$\mathcal{C} \downarrow$	graph	labels	$\mathcal{C} \downarrow$	graph	labels	$\mathcal{C} \downarrow$	graph	labels	
MinCut	23.3	70.3	35.8	14.1	78.9	25.9	29.6	63.1	25.4	22.7	70.5	64.6	27.8	64.3	48.3
Ortho	28.0	65.6	38.4	18.4	74.5	26.1	57.8	32.9	20.3	27.8	65.7	64.6	33.0	59.5	44.7

cluster produces a trivial locally optimal solution that traps gradient-based optimization methods. MinCutPool addresses this problem by adapting spectral orthogonality constraint in the form of soft-orthogonality regularization $\|\mathbf{C}^\top \mathbf{C} - \mathbf{I}\|_F$ from Bansal et al. (2018).

We find that, interestingly, MinCutPool’s orthogonality constraint is overly restrictive when combined with softmax class assignment. This can be seen empirically by tracking the progress of MinCutPool optimization, shown in Figure 1. We observe that after 200 epochs on the CORA dataset, MinCutPool’s soft orthogonality regularization term dominates its clustering term, which becomes *worse than random* after training. We hypothesize that orthogonalizing softmax representations may be a difficult regularization objective to optimize.

Another effect of MinCutPool’s harsh orthogonality constraint is seen in Table 2. We show that a constraint-only version (termed "Ortho") that we implemented actually performs better on the downstream classification task than the original loss on all datasets except PubMed, while performing uniformly worse with respect to the graph metrics – as expected, since it ignores the clustering loss. This suggests that the constraint is not well-designed for the task, and may be trapping the method in local minima. We further corroborate this in Section 5.2.

We propose *collapse regularization*, a more relaxed constraint, that prevents the trivial partition while not dominating the optimization of the main objective. The regularizer is a Frobenius norm of the (soft) cluster membership counts, normalized to range $[0, \sqrt{k}]$. It gets value of 0 when cluster sizes are perfectly balanced, and \sqrt{k} in the case all clusters collapse to one. We also improve the training by applying dropout (Srivastava et al., 2014) to GNN representations before the softmax, preventing the gradient descent from getting stuck in the local optima of the highly non-convex objective function.

4.3 Theoretical insights

In this section we examine the validity of our proposed objective function, and in particular our novel collapse constraint. Specifically, we show two theorems:

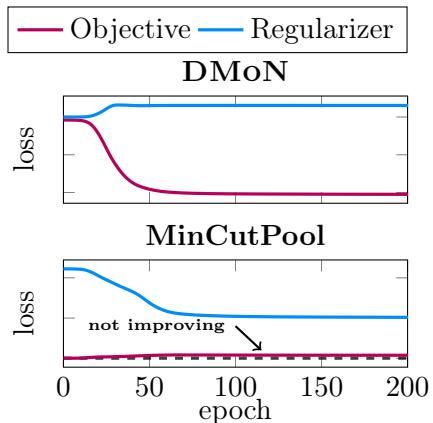


Figure 1: Optimization progress of MinCut and DMoN on Cora dataset. MinCut optimizes the regularizer, while DMoN minimizes its main objective. Dashed line shows the initial loss value.

1. We prove that optimizing our collapse constraint avoids the solution with trivial clusters, and hence accomplishes the effect for which it was designed.
2. Since the collapse constraint inherently discourages unbalanced cluster assignments (to avoid the trivial clustering), we also show that this discouragement does not affect the asymptotic performance of our overall objective function.

The first theorem essentially tells us that the collapse regularization can not hurt optimization in non-trivial graphs:

Theorem 1 *If the clustering solution encoded by the C matrix has positive modularity, then our objective score $\mathcal{L}_{\text{DMoN}}$ is smaller than the one produced by the trivial clustering.*

Proof We show that both components of the loss are smaller. Since modularity of the trivial clustering is 0, any positive modularity will result in a smaller loss. Maximum of L_2 norm of a vector with given L_1 norm is achieved when only one element is nonzero – in case of the collapse regularization, when all nodes are in one cluster. Therefore, any other partition produces a smaller value. ■

Our second theorem is based on previous work on the “consistency” of graph clustering algorithms under an appropriate graph generative model. In Bickel and Chen (2009) and Zhao et al. (2012), a graph clustering algorithm is consistent if the global maximum clustering solution from its objective function has (in-probability) vanishing error rate in the limit of large graphs, when graphs are generated from the cluster-laden Degree-Corrected Stochastic Block Model (DC-SBM). We now detail this model and specific notions of consistency, and show that our novel, GNN-compatible objective function has consistency properties that are equivalent to standard objective functions.

The standard (degree-free) Stochastic Block Model (Nowicki and Snijders, 2001) is a generative graph model which divides n vertices into k classes, and then places edges between two vertices v_i and v_j with probability p_{ij} determined from the class assignments. Specifically, each vertex v_i is given a class $y_i \in \{1, \dots, k\}$, and an edge $\{v_i, v_j\}$ is added to the edge set E with probability $P_{y_i y_j}$, where P is a symmetric $k \times k$ matrix containing the between/within-community edge probabilities. In other words, each entry \mathbf{A}_{ij} of the adjacency matrix is an independent Bernoulli random variable with probability $P_{y_i y_j} = \mathbb{E}[\mathbf{A}_{ij}]$. Assortative clustering structure in a graph can be induced using the SBM by setting the on-diagonal probabilities of P higher than the off-diagonal probabilities. Thus, the SBM has been used to analyze graph clustering algorithms both theoretically (Bickel and Chen, 2009; Abbe and Sandon, 2015) and empirically (Mothe et al., 2017).

The Degree-Corrected SBM (Karrer and Newman, 2011) was introduced to handle the heterogeneity of real-world degree distributions. In the DC-SBM, each node v_i is given a degree parameter θ_i , and edges are generated via the formula $\mathbb{E}[\mathbf{A}_{ij}] = \theta_i \theta_j P_{y_i y_j}$. Thus two nodes with higher degree parameters will connect more frequently under this model.

In this paper, we adopt the theoretical setting from Zhao et al. (2012): in a DC-SBM \mathbb{P}_n on n nodes with k classes, each node v_i is given a label/degree pair (y_i, θ_i) drawn from a discrete joint distribution $\Pi_{k \times m}$ which is fixed and does not depend on n . This implies that each θ_i is one of a fixed set of values $0 \leq x_1 \leq \dots \leq x_m$. To facilitate analysis of asymptotic

graph sparsity, we parameterize the edge probability matrix P as $P_n = \rho_n P$ where P is independent of n , and $\rho_n = \lambda_n/n$ where λ_n is the average degree of the network. We assume that $x_m^2 \max P_n \leq 1.0$ so that all edge probabilities are proper. We define the marginal distribution of the labels y as $\tilde{\pi} = (\tilde{\pi}_1, \dots, \tilde{\pi}_k)$.

Given a DC-SBM \mathbb{P}_n on n nodes with k classes, we denote $\hat{\mathbf{y}}_n = [\hat{y}_1, \hat{y}_2, \dots, \hat{y}_n]$ to be a predicted class label vector, and $\hat{\mathbf{C}}_n$ to be the associated $n \times k$ row-wise one-hot matrix. With $\mathbf{A}^{(n)}$ the random adjacency matrix of the DC-SBM, define the random variable $\hat{\mathbf{C}}_n^*$ to be the global maximum of $\mathcal{L}_{\text{DMoN}}(\cdot; \mathbf{A}^{(n)})$ over all $n \times k$ label matrices. Let \mathbf{C}_n be the true one-hot label matrix associated with the DC-SBM class labels $\mathbf{y}_n := [y_1, y_2, \dots, y_n]$. Let μ be any $k \times k$ permutation matrix, and let $\|\cdot\|_F$ be the Frobenius norm. As in Zhao et al. (2012), we consider two notions of consistency:

$$\mathbb{P}_n \left[\min_{\mu} \|\hat{\mathbf{C}}_n^* \mu - \mathbf{C}_n\|_F^2 = 0 \right] \rightarrow 1 \quad \text{as } n \rightarrow \infty \quad (\text{strong consistency}) \quad (6)$$

$$\forall \epsilon > 0, \quad \mathbb{P}_n \left[\min_{\mu} \frac{1}{n} \|\hat{\mathbf{C}}_n^* \mu - \mathbf{C}_n\|_F^2 < \epsilon \right] \rightarrow 1 \quad \text{as } n \rightarrow \infty \quad (\text{weak consistency}) \quad (7)$$

We now state our main theorem, which shows that DMoN's novel collapse constraint does not prevent asymptotically-consistent prediction of the optimal class labels.

Theorem 2 *Under the conditions of Theorem 3.1 from Zhao et al. (2012), when the cluster sizes of the DC-SBM are equal, $\mathcal{L}_{\text{DMoN}}$ is strongly consistent when $\lambda_n/\log(n) \rightarrow \infty$ and weakly consistent when $\lambda_n \rightarrow \infty$.*

Proof This result relies on a general theorem from Zhao et al. (2012), which is restated here for completeness. For convenience, dependence on n will be ignored when it does not cause ambiguity. The DMoN objective can be re-written as a function of inter-class edge counts $O(\mathbf{A}, y)_{rs} = \sum_{i,j} \mathbf{A}_{ij} \cdot 1(y_i = r, y_j = s)$ and class proportions $\pi(y)_r = n^{-1} \sum_i 1(y_i = r)$. Let $\bar{O}(\mathbf{A}, y)_r = \sum_s O(\mathbf{A}, y)_{rs}$ and $m(\mathbf{A}) = \sum_r \bar{O}(\mathbf{A}, y)_r$. Henceforth dependence on y will sometimes be ignored for clarity. With these notations, we have

$$\mathcal{L}_{\text{DMoN}}(C; \mathbf{A}) = -\frac{1}{2m(\mathbf{A})} \sum_r \left[O(\mathbf{A}, y)_{rr} - \frac{\bar{O}(\mathbf{A}, y)_r^2}{m(\mathbf{A})} \right] + \frac{\sqrt{k}}{n} \|n\pi(y)\|_F - 1 \quad (8)$$

Hence $\mathcal{L}_{\text{DMoN}}(C; \mathbf{A})$ can be written as a function $F(O, \pi)$ of a $k \times k$ edge count matrix O and a $k \times 1$ class proportion vector π . Theorem 4.1 from Zhao et al. (2012) shows that if F satisfies regularity conditions and is uniquely minimized by the conditional expectation of its arguments under the DC-SBM \mathbb{P}_n , then the objective function that F describes is both strongly and weakly consistent under the model.

The regularity conditions are listed in Section 4 of Zhao et al. (2012) and are easy to verify against $\mathcal{L}_{\text{DMoN}}$. We now show that $\mathcal{L}_{\text{DMoN}}$ is uniquely minimized by the conditional expectation of its arguments under the DC-SBM \mathbb{P}_n . In other words, $\mathcal{L}_{\text{DMoN}}$ must be uniquely minimized at any point (y^*, \mathbf{A}^*) such that $\mathbb{E}_n[\pi(y_n)] = \pi(y^*)$ and $\mathbb{E}_n[\mathbf{A}^{(n)}] = \mathbf{A}^*$. Note that the first term of $\mathcal{L}_{\text{DMoN}}$ is the (negative) standard Newman-Girvan modularity, which was shown to be minimized under the expected DC-SBM in Zhao et al. (2012) (see the proof of Theorem 3.1). The proof is complete as the DMoN collapse constraint is minimized when the cluster sizes are equal. \blacksquare

5. Experiments

In this section, we describe empirical experiments—Involving both synthetic and real-world data—on DMON and baseline methods, to test robustness and performance against both graph clustering and label alignment metrics. We use open-source graph simulation tools, publicly-available datasets, and we release the implementation of DMON at this URL¹.

Datasets. We use 11 real-world datasets for assessing model quality. Cora, Citeseer, and Pubmed (Sen et al., 2008) are citation networks; nodes represent papers connected by citation edges; features are bag-of-word abstracts, and labels represent paper topics. Amazon PC and Amazon Photo (Shchur et al., 2018) are subsets of the Amazon co-purchase graph for the computers and photo sections of the website, where nodes represent goods with edges between ones frequently purchased together; node features are bag-of-word reviews, and class labels are product category. Coauthor CS, Coauthor Phys, Coauthor Med, Coauthor Chem, and Coauthor Eng (Shchur et al., 2018; Shchur and Günnemann, 2019) are co-authorship networks based on the Microsoft Academic Graph for the computer science, physics, medicine, and engineering fields respectively; nodes are authors, which are connected by edge if they co-authored a paper together; node features are a collection of paper keywords for author’s papers; class labels indicate most common fields of study. OGB-arXiv (Hu et al., 2020) is a paper co-citation dataset based on arXiv papers indexed by the Microsoft Academic graph. Nodes are papers; edges are citations, and class labels indicate the main category of the paper.

Baselines. As we study how to leverage the information from both the graph and attributes, we employ two baselines that employ either strictly graph or attribute information. We give a brief description of the baselines used below:

- **k-means(features)** is our baseline that only considers the feature data. We use the local Lloyd algorithm (Lloyd, 1982) with the k-means++ seeding strategy (Arthur and Vassilvitskii, 2007).
- **SBM** (Peixoto, 2014a) is a baseline that only relies on the graph structure. We estimate a constrained stochastic block model with given k , the maximum number of clusters.
- **k-means(DeepWalk, features)** (Perozzi et al., 2014b) represents a naïve strategy of concatenating node attributes to learned node embeddings of the graph without attributes.
- **k-means(DGI)** (Veličković et al., 2019) demonstrates the need of joint learning of clusters and representations. We learn unsupervised node representations of the attributed graph with DGI and run k-means on the resulting representations.
- **AGC(graph, features)** (Zhang et al., 2019) applied spectral clustering over features smoothed by the graph.
- **DAEGC(graph, features)** (Wang et al., 2019a) is a graph reconstruction-based clustering method.

Figure 2: Dataset statistics.

dataset	$ V $	$ E $	$ X $	$ Y $
Cora	2708	5278	1433	7
Citeseer	3327	4614	3703	6
Pubmed	19717	44325	500	3
Amazon PC	13752	143604	767	10
Amazon Photo	7650	71831	745	8
Coauthor Eng	14927	49305	4839	16
Coauthor CS	18333	81894	6805	15
Coauthor Phys	34493	247962	8415	5
Coauthor Chem	35409	157358	4877	14
Coauthor Med	63282	810314	5538	17
OGB-arXiv	169343	583121	128	40

¹ https://github.com/google-research/google-research/tree/master/graph_embedding/dmon

- **SDCN(graph, features)** (Bo et al., 2020) is another graph reconstruction-based clustering method. Representations are initialized through k-means over auto-encoded features.
- **NOCD(graph, features)** (Shchur and Günnemann, 2019) directly optimizes negative log-likelihood of the graph reconstruction of the Bernoulli-Poisson model.
- **DiffPool(graph, features)** (Ying et al., 2018b) is an early graph pooling method.
- **MinCutPool(graph, features)** (Bianchi et al., 2020) is a deep pooling method that orthogonalizes the cluster representations (cf. Section 4 for discussion).
- **Ortho(graph, features)** (Bianchi et al., 2020) is a variant of **MinCutPool** that only does the cluster orthogonalization without any graph-related objective.

Metrics. We measure both the graph-based metrics of clustering and label correlation to study clustering performance of attributed graphs both in terms of graph and attribute structure. For experiments on real-world data, we measure both (1) standard graph-based clustering metrics, and (2) correlation to ground-truth node labels. Our graph-based metrics are average cluster conductance (as per definition from Yang and Leskovec (2015)) and graph modularity (Newman, 2006a). Our label-based metrics are normalized mutual information (NMI) between the cluster assignments and labels and pairwise F1 score between all node pairs and their associated cluster pairs. For experiments on synthetic data, we report only the NMI against the (simulated) cluster labels. Where possible, we normalize all metrics by multiplying them by 100 for readability and ease of comparison.

Parameter settings. We run both synthetic and real-world experiments for 10 times and average results across runs. All models were implemented in Tensorflow 2 and trained on CPUs. We fix the architecture for all GNNs (including DMoN) have one hidden layer—with 512 neurons for real-world data experiments, and 64 neurons for experiments with smaller synthetic graphs. We set the maximum number of clusters to 16 for all datasets and methods.

We aim to keep the hyperparameters constant across datasets, since (i) tuning hyperparameters in an unbiased way is non-trivial in the unsupervised setting, (ii) we report an array of metrics that relate to the graph and to the labels – favoring one group over other is not in the scope of this paper. One notable exception is SDCN, as it requires an autoencoder on the features to function. We keep the original three-layer encoder structure and set the architecture to [32, 32, 128, 128, 32, 32] for synthetic SBM graphs and [500, 500, 2000, 2000, 500, 500] for all other datasets. Setting hyperparameters in a more fair way to keep the total number of parameters same across different methods led to random performance.

We now discuss method-specific hyperparameters for methods that have specific settings:

- **k-means(DeepWalk, features)**: We keep the learning parameters as per Perozzi et al. (2014b) number of walks $\gamma = 80$, walk length $t = 80$, and window size $w = 10$.
- **AGC(graph, features)**: We set $k = 1$ to mimic a single-layer graph convolution that we set for all GNN-based methods.
- **DAEGC(graph, features)**: We set the clustering loss coefficient $\gamma = 10$ as per the original paper.
- **SDCN(graph, features)**: We set the clustering loss coefficient $\alpha = 0.1$ and GNN loss coefficient $\beta = 0.01$ as per the original paper.
- **NOCD(graph, features)**: We set the dropout to 0.5 uniformly across datasets and set the batch size to 2000 as per the original paper.
- **DMoN (graph, features)**: We set the dropout to 0.5 uniformly across datasets.

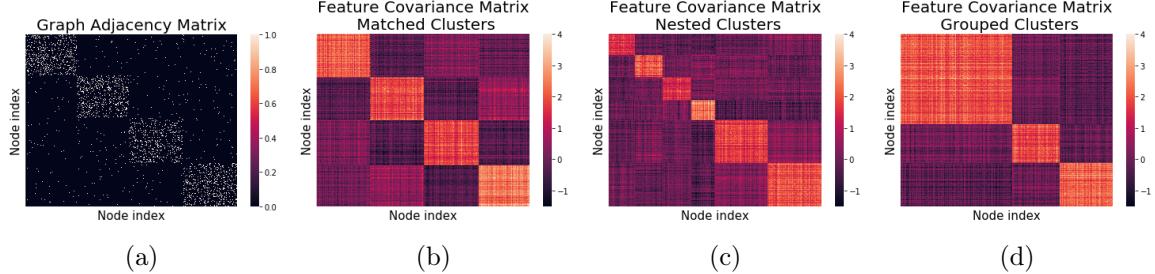


Figure 3: **Illustration of synthetic data.** (a) 4-cluster graph adjacency matrix. (b) Covariance matrix of “matched” features: features that are clustered according to the graph clusters. (c) Covariance matrix of “nested” features: features that are clustered by incomplete nesting of the graph clusters. (d) Covariance matrix of “grouped” features: features that are clustered by incomplete grouping of the graph clusters.

The dropout setting we use does not benefit DAEC, SDCN, DiffPool, or MinCutPool, hence we are not able to unify the setting of that parameter across methods.

5.1 Simulation Experiments on Stochastic Block Model with Features

To explore the robustness and sensitivity of DMON and baselines to variance in the graph and node features, we conduct a study on synthetic graphs using an attributed, degree-corrected stochastic block model (ADC-SBM). The SBM (Snijders and Nowicki, 1997) plants a partition of clusters (“blocks”) in a graph, and generates edges via a distribution conditional on that partition. This model has been used extensively to benchmark graph clustering methods (Fortunato and Hric, 2016), and has recently been used for experiments on state-of-the-art supervised GNNs (Dwivedi et al., 2023). In our version of the model, node features are also generated, using a multivariate mixture model, with the mixture memberships having some association (or de-association) with the cluster memberships. We proceed to describing the graph generation and feature generation components of our model.

ADC-SBM graph generation. We fix a number of nodes n and a number of clusters k , and choose node cluster memberships uniformly-at-random. Define the matrix $\mathbf{D}_{k \times k}$ where \mathbf{D}_{ij} is the expected number of edges shared between nodes in clusters i and j . We determine \mathbf{D} by fixing (1) the expected average degree of the nodes $d \in \{1, n\}$, and (2) the expected average sub-degree $d_{out} \leq d$ of a node to any cluster other than its own. Note that the difference $d_{in} - d_{out}$, where $d_{in} := d - d_{out}$, controls the spectral detectability of the clusters (Nadakuditi and Newman, 2012). Finally, we generate a power-law n -vector θ , where θ_i is proportional to i ’s expected degree. We use the generated memberships and the generated parameters \mathbf{D} and θ as inputs to the degree-corrected SBM function from the `graph-tool` (Peixoto, 2014b) package.

ADC-SBM feature generation. We generate feature memberships from k_f cluster labels. For graph clustering GNNs that operate both on edges and node features, it is important to examine performance on data where feature clusters diverge from or segment the graph clusters: thus potentially $k_f \neq k$. We examine cases where feature memberships match, group, or nest the graph memberships, as illustrated in Figure 3. With feature memberships

in-hand, we generate k zero-mean feature cluster centers from a s -multivariate normal with covariance matrix $\sigma_c^2 \cdot \mathbf{I}_{s \times s}$. Then, for feature cluster $i \leq k_f$, we generate its features from a s -multivariate normal with covariance matrix $\sigma^2 \cdot \mathbf{I}_{s \times s}$. Note that the ratio σ_c^2/σ^2 controls the expected value of the classical between/within-sum-of-squares of the clusters.

The above paragraphs describe a single generation of our synthetic benchmark model, the ADC-SBM. To study model robustness, we define “default” ADC-SBM parameters, and explore model parameters in a range around the defaults. We configure our default model as follows: we generate graphs with $n = 1,000$ nodes grouped in $k = 4$ clusters, and $s = 32$ -dimensional features grouped in $k_f = 4$ matching feature clusters with $\sigma = 1$ intra-cluster center variance and $\sigma_c = 3$ cluster center variance. We mimic real-world graphs’ degree distribution with $d = 20$ average degree and $d_{out} = 2$ average inter-cluster degree with power law parameters $d_{min} = 2, d_{max} = 4, \alpha = 2$. In total, we consider 6 different scenarios, as described in Table 3.

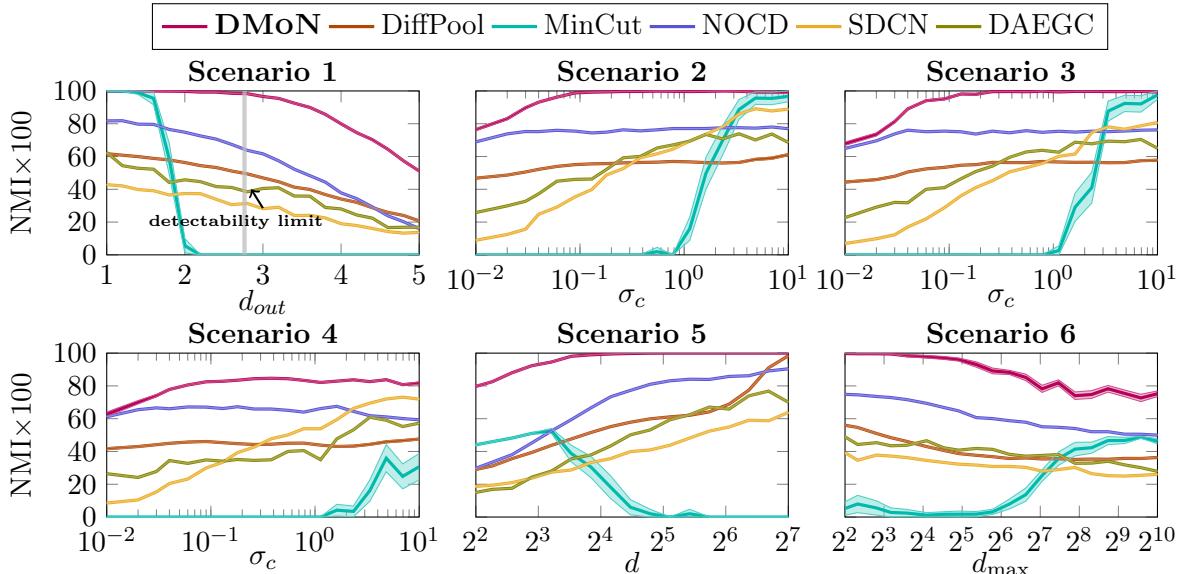


Figure 4: Synthetic results on the ADC-SBM model with 6 different scenarios described in Table 3. DMoN significantly outperforms other neural graph pooling method baselines.

Table 3: Synthetic ADC-SBM benchmark scenarios.

Scenario	Parameter	Description
1	$d_{out} \in [2.0, 5.0]$	Increase graph cluster mixing signal. Higher = weaker clusters.
2	$\sigma_c \in [10^{-2}, 10^1]$	Increase feature cluster center variance. Higher = stronger clusters.
3	$\sigma_c \in [10^{-2}, 10^1]$	Increase feature cluster center variance, with nested feature clusters.
4	$\sigma_c \in [10^{-2}, 10^1]$	Increase feature cluster center variance, with grouped feature clusters.
5	$d \in [2^2, 2^7]$	Increase average degree. Higher = clearer graph signal.
6	$d_{max} \in [2^2, 2^{10}]$	Increase power law upper-bound. Higher = more extreme power law.

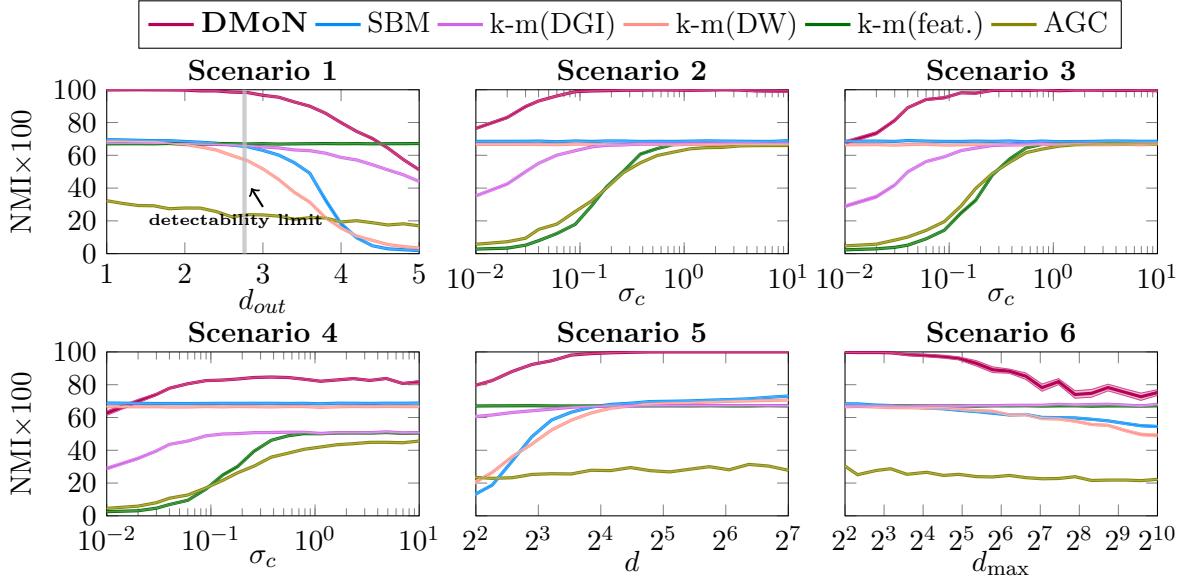


Figure 5: Synthetic results on the ADC-SBM model with 6 different scenarios described in Table 3. DMoN leverages information from both graph structure and node attributes.

Results. We split the presentation of results for neural methods (Figure 4) and other baselines (Figure 5) for the ease of understanding. Overall, end-to-end graph pooling methods outperform multi-step baselines by a wide margin. DMoN demonstrates overwhelming superiority over all baselines, with the most significant improvements over other pooling methods. MinCut pooling suffers from the presence of even the weak noise in the graph (Scenario 1) or in the features (Scenario 2). Moreover, it is susceptible to both nested and grouped features (Scenarios 3 and 4), while DMoN and DiffPool are less sensitive to these variations. Both DiffPool and MinCutPool are dependent on the sparsity level and degree homogeneity of the graph — DiffPool performs better on denser graphs while MinCutPool shows the opposite picture.

To better understand the limits of the task, we study the performance of our baselines and report the results on Figure 5. In particular our interest lies in the performance of the SBM and pure k-means over features, as these two baselines depict the performance possible when utilizing only one aspect of the data. Scenario 1 shows that DMoN can effectively leverage the feature signal to obtain outstanding clustering performance even when the graph structure is close to random, far beyond the spectral detectability threshold (pictured in gray). Scenario 2 demonstrates that even in the presence of a weak feature signal DMoN outperforms stochastic SBM minimization. We also notice that while the k-means(DGI) baseline offers some improvements over using features or the graph structure alone, it never surpasses the strongest signal provider in the graph, never being better than the best one between k-means(features) and SBM. K-means applied over the extracted DeepWalk representations are also almost never stronger than the community detection using direct SBM likelihood optimization.

Table 4: Results on three datasets from Sen et al. (2008) in terms of graph conductance \mathcal{C} , modularity \mathcal{Q} , NMI with ground-truth labels, and pairwise F1 measure. We group the methods into three categories: baselines using only one aspect of data, neural representation learning, and neural graph pooling methods. We highlight best neural method performance.

method	Cora				Citeseer				Pubmed					
	graph		labels		graph		labels		graph		labels			
	$\mathcal{C} \downarrow$	$\mathcal{Q} \uparrow$	NMI \uparrow	F1 \uparrow		$\mathcal{C} \downarrow$	$\mathcal{Q} \uparrow$	NMI \uparrow	F1 \uparrow		$\mathcal{C} \downarrow$	$\mathcal{Q} \uparrow$	NMI \uparrow	F1 \uparrow
k-m(feat)	61.7	19.8	18.5	27.0	60.5	30.3	24.5	29.2	55.8	33.4	19.4	24.4		
SBM	15.4	77.3	36.2	30.2	14.2	78.1	15.3	19.1	39.0	53.5	16.4	16.7		
k-m(DW)	62.1	30.7	24.3	24.8	68.1	24.3	27.6	24.8	16.6	75.3	22.9	17.2		
AGC	48.9	43.2	34.1	28.9	41.9	50.0	25.5	27.5	44.9	46.8	18.2	18.4		
SDCN	37.5	50.8	27.9	29.9	20.0	62.3	31.4	41.9	22.4	50.3	19.5	29.9		
DAEGC	56.8	33.5	8.3	13.6	47.6	36.4	4.3	18.0	53.6	37.5	4.4	11.6		
k-m(DGI)	28.0	64.0	52.7	40.1	17.5	73.7	40.4	39.4	82.9	9.6	22.0	26.4		
NOCD	14.7	78.3	46.3	36.7	6.8	84.4	20.0	24.1	21.7	69.6	25.5	20.8		
DiffPool	26.1	66.3	32.9	34.4	26.0	63.4	20.0	23.5	32.9	56.8	20.2	26.3		
MinCut	23.3	70.3	35.8	25.0	14.1	78.9	25.9	20.1	29.6	63.1	25.4	15.8		
Ortho	28.0	65.6	38.4	26.6	18.4	74.5	26.1	20.5	57.8	32.9	20.3	13.9		
DMoN	12.2	76.5	48.8	48.8	5.1	79.3	33.7	43.2	17.7	65.4	29.8	33.9		

Table 5: Results on four datasets from Shchur et al. (2018) in terms of graph conductance \mathcal{C} , modularity \mathcal{Q} , NMI with ground-truth labels, and pairwise F1 measure. We group the methods into three categories: baselines using only one aspect of data, neural representation learning, and neural graph pooling methods. We highlight best neural method performance.

method	Amazon PC				Amazon Photo				Coauthor CS				Coauthor Eng							
	graph		labels		graph		labels		graph		labels		graph		labels					
	$\mathcal{C} \downarrow$	$\mathcal{Q} \uparrow$	NMI \uparrow	F1 \uparrow		$\mathcal{C} \downarrow$	$\mathcal{Q} \uparrow$	NMI \uparrow	F1 \uparrow		$\mathcal{C} \downarrow$	$\mathcal{Q} \uparrow$	NMI \uparrow	F1 \uparrow		$\mathcal{C} \downarrow$	$\mathcal{Q} \uparrow$	NMI \uparrow	F1 \uparrow	
k-m(feat)	84.5	5.4	21.1	19.2	79.6	10.5	28.8	19.5	49.1	23.1	35.7	39.4	42.7	27.1	24.5	32.5				
SBM	31.0	60.8	48.4	34.6	18.6	72.7	59.3	47.4	20.3	72.7	58.0	47.7	15.8	77.0	33.3	27.5				
k-m(DW)	67.6	11.8	38.2	22.7	60.6	22.9	49.4	33.8	33.1	59.4	72.7	61.2	5.7	67.4	47.7	50.0				
AGC	43.2	42.8	51.3	35.3	33.8	55.9	59.0	44.2	41.5	40.1	43.3	31.9	32.3	46.4	30.8	31.2				
DAEGC	39.0	43.3	42.5	37.3	19.3	58.0	47.6	45.0	39.4	49.1	36.3	32.4	31.9	50.9	12.5	26.1				
SDCN	25.1	45.6	24.9	45.2	19.7	53.3	41.7	45.1	33.0	55.7	59.3	54.7	21.8	64.6	45.3	45.9				
k-m(DGI)	61.9	22.8	22.6	15.0	51.5	35.1	33.4	23.6	35.1	57.8	64.6	51.9	29.3	60.4	49.7	37.2				
NOCD	26.4	59.0	44.8	37.8	13.7	70.1	62.3	60.2	20.9	72.2	70.5	56.4	16.0	75.6	50.7	35.4				
DiffPool	35.6	30.4	22.1	38.3	26.5	46.8	35.9	41.8	33.6	59.3	41.6	34.4	34.9	55.0	22.0	21.8				
MinCut	<i>did not converge</i>				<i>22.7</i>				<i>70.5</i>	<i>64.6</i>	<i>47.8</i>	<i>18.2</i>	<i>74.8</i>	<i>42.4</i>	<i>27.8</i>					
Ortho	<i>did not converge</i>				<i>27.8</i>				<i>65.7</i>	<i>64.6</i>	<i>46.1</i>	<i>18.1</i>	<i>74.8</i>	<i>43.2</i>	<i>28.1</i>					
DMoN	18.0	59.0	49.3	45.4	12.7	70.1	63.3	61.0	17.5	72.4	69.1	59.8	7.6	72.1	51.8	55.3				

5.2 Real-world Data

We now move on to studies on real-world networks, featuring DMoN and 7 baselines on 7 different datasets. DMoN achieves better clustering performance than its neural counterparts

on every single dataset and metric besides losing twice to DGI+k-means on Cora and Citeseer in terms of NMI. Compared to SBM, a method that exclusively optimizes modularity, we are able to stay within 3% in terms of modularity, while simultaneously clustering the features. Surprisingly, on Citeseer and Pubmed we achieve better modularity than the method optimizing it directly — we attribute that to very high correlation between graph structure and the features.

Compared to other pooling methods, DMoN improves by over 40% in conductance, modularity and NMI on average. In particular, DiffPool achieves overall poor performance across metrics due to its quadratic reconstruction term that does not scale well with graph sparsity (cf. Scenario 5). MinCut pooling performs better, but only manages to match the performance of non-pooling neural representation learning methods on one dataset in terms of ground-truth label NMI. On Amazon PC and Amazon Photo both MinCutPool and its orthogonality-only version failed to converge, even with tuning the parameters. We attribute that to extremely uneven structure of these graphs, as popular products are co-purchased with a lot of other items, so the effects discussed in Epasto et al. (2017); Leskovec et al. (2008) are prohibiting good cuts. This corresponds to high values of d ad d_{\max} in our synthetic scenarios 5 and 6. We also highlight that we beat MinCutPool in terms of conductance (average graph cut) on all datasets, even though it attempts to optimize for this metric.

On large-scale datasets presented in Table 6, DMoN achieves 16% better performance on average across all metrics compared to other end-to-end learning methods. Some methods, like AGC, DAEGC, and DiffPool are not able to scale to large-scale graphs due to their quadratic time complexity.

Overall, DMoN demonstrates excellent performance on both graph clustering and label correlation, successfully leveraging both graph and attribute information. Both synthetic and real-world experiments prove that DMoN is vastly superior to its counterparts in attributed graph clustering.

Table 6: Results on four large-scale datasets in terms of graph conductance \mathcal{C} , modularity \mathcal{Q} , NMI with ground-truth labels, and pairwise F1 measure. We group the methods into three categories: baselines using only one aspect of data, neural representation learning, and neural graph pooling methods. We highlight best neural method performance.

method	Coauthor Phys				Coauthor Chem				Coauthor Med				OGB-arXiv			
	<u>graph</u>	<u>labels</u>	<u>graph</u>	<u>labels</u>	<u>graph</u>	<u>labels</u>	<u>graph</u>	<u>labels</u>	<u>graph</u>	<u>labels</u>	<u>graph</u>	<u>labels</u>	<u>graph</u>	<u>labels</u>	<u>graph</u>	<u>labels</u>
k-m(feat)	57.0	19.4	30.6	42.9	42.9	18.2	13.9	35.1	54.7	19.3	11.8	31.7	75.9	16.4	20.3	20.2
SBM	25.9	66.9	45.4	30.4	18.4	74.6	25.4	25.0	21.1	72.0	36.1	31.1	24.8	67.6	31.9	28.3
k-m(DW)	44.7	47.0	43.5	24.3	14.0	74.8	36.5	33.8	16.6	72.1	43.1	39.4	29.5	58.2	28.4	36.0
SDCN	32.1	52.8	50.4	39.9	29.9	58.7	33.3	32.8	34.8	54.2	25.2	26.5	31.2	36.8	15.3	28.8
k-m(DGI)	38.6	51.2	51.0	30.6	31.6	60.6	40.8	32.9	35.7	56.5	34.8	27.7	58.7	29.7	30.0	24.6
NOCD	25.7	65.5	51.9	28.7	19.2	73.1	43.1	40.1	22.0	69.7	42.5	37.6	41.1	41.9	20.7	38.2
MinCut	27.8	64.3	48.3	24.9	21.2	72.2	39.0	32.0	22.8	69.7	40.0	32.1	37.4	52.6	36.0	27.1
Ortho	33.0	59.5	44.7	23.7	21.6	71.8	38.5	31.0	22.9	69.5	40.4	32.1	37.8	52.2	35.6	26.7
DMoN	18.8	65.8	56.7	42.4	16.2	73.9	43.2	43.7	17.6	71.6	43.4	40.0	22.3	57.4	37.6	45.7

6. Conclusion

In this work, we study GNN pooling through the lens of attributed graph clustering. We introduce Deep Modularity Networks (DMON), an unsupervised objective and realize it with a GNN which can recover high quality clusters. We compare against challenging baselines that baselines that optimize structure (SBM), features (kmeans), or both (DGI+k-means), in addition to a recently proposed state-of-the-art pooling method (MinCutPool).

We explore the limits of GNN clustering methods in terms of both graph and feature signals using synthetic data, where we see that DMON better leverages structure and attributes than all existing methods. In extensive experiments on real datasets we show that the clusters found by DMON are more likely to correspond to ground truth labels, and have better properties as illustrated by clustering metrics (e.g. conductance or modularity). We hope that this work will further advancements in unsupervised learning for GNNs as well as attributed graph clustering, allowing further advances in graph learning.

References

- Emmanuel Abbe and Colin Sandon. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In FOCS. IEEE, 2015.
- David Arthur and Sergei Vassilvitskii. K-means++: The advantages of careful seeding. In SODA, 2007.
- Nitin Bansal, Xiaohan Chen, and Zhangyang Wang. Can we gain more from orthogonality regularizations in training deep CNNs? In NIPS, 2018.
- Mohamed Ishmael Belghazi, Aristide Baratin, Sai Rajeswar, Sherjil Ozair, Yoshua Bengio, Aaron Courville, and R Devon Hjelm. Mine: mutual information neural estimation. In ICML, 2018.
- Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. Spectral clustering with graph neural networks for graph pooling. In ICML, 2020.
- Peter J Bickel and Aiyou Chen. A nonparametric view of network models and newman–girvan and other modularities. Proceedings of the National Academy of Sciences, 2009.
- Deyu Bo, Xiao Wang, Chuan Shi, Meiqi Zhu, Emiao Lu, and Peng Cui. Structural deep clustering network. In Proceedings of The Web Conference 2020, pages 1400–1410, 2020.
- Ulrik Brandes, Daniel Delling, Marco Gaertler, Robert Görke, Martin Hoefer, Zoran Nikoloski, and Dorothea Wagner. Maximizing modularity is hard. arXiv preprint physics/0608255, 2006.
- Michael M Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond euclidean data. IEEE Signal Processing Magazine, 2017.
- 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), pages 584–589. IEEE, 2016.

- Ines Chami, Sami Abu-El-Haija, Bryan Perozzi, Christopher Ré, and Kevin Murphy. Machine learning on graphs: A model and comprehensive taxonomy. *JMLR*, 2022.
- Haochen Chen, Bryan Perozzi, Yifan Hu, and Steven Skiena. HARP: Hierarchical representation learning for networks. In *AAAI*, 2018.
- Stéphan Cléménçon, Hector De Araozza, Fabrice Rossi, and Viet Chi Tran. Hierarchical clustering for graph visualization. *arXiv preprint arXiv:1210.5693*, 2012.
- Weiwei Cui, Hong Zhou, Huamin Qu, Pak Chung Wong, and Xiaoming Li. Geometry-based edge clustering for graph visualization. *IEEE Transactions on Visualization and Computer Graphics*, 14(6):1277–1284, 2008.
- Michaël Defferrard, Xavier Bresson, and Pierre Vandergheynst. Convolutional neural networks on graphs with fast localized spectral filtering. In *NIPS*, 2016.
- Chenhui Deng, Zhiqiang Zhao, Yongyu Wang, Zhiru Zhang, and Zhuo Feng. GraphZoom: A multi-level spectral approach for accurate and scalable graph embedding. In *ICLR*, 2020.
- David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *NIPS*, 2015.
- Vijay Prakash Dwivedi, Chaitanya K. Joshi, Anh Tuan Luu, Thomas Laurent, Yoshua Bengio, and Xavier Bresson. Benchmarking graph neural networks. *Journal of Machine Learning Research*, 24(43):1–48, 2023. URL <http://jmlr.org/papers/v24/22-0567.html>.
- Alessandro Epasto, Silvio Lattanzi, and Renato Paes Leme. Ego-splitting framework: From non-overlapping to overlapping clusters. In *KDD*, 2017.
- Miroslav Fiedler. Algebraic connectivity of graphs. *Czechoslovak mathematical journal*, 1973.
- Santo Fortunato and Darko Hric. Community detection in networks: A user guide. *Physics reports*, 2016.
- Hongyang Gao and Shuiwang Ji. Graph U-nets. In *ICML*, 2019.
- Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *ICML*, 2017.
- Benjamin H Good, Yves-Alexandre De Montjoye, and Aaron Clauset. Performance of modularity maximization in practical contexts. *Physical Review E*, 81(4):046106, 2010.
- Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *KDD*, 2016.
- Michael Gutmann and Aapo Hyvärinen. Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *AISTATS*, 2010.

- William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 2017.
- 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. In *ICLR*, 2019.
- 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. *NeurIPS*, 2020.
- Weihua Hu*, Bowen Liu*, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *ICLR*, 2020.
- Brian Karrer and Mark EJ Newman. Stochastic blockmodels and community structure in networks. *Physical review E*, 83(1):016107, 2011.
- Brian W Kernighan and Shen Lin. An efficient heuristic procedure for partitioning graphs. *The Bell system technical journal*, 1970.
- Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *ICLR*, 2017.
- Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In *NIPS*, 2017.
- Junhyun Lee, Inyeop Lee, and Jaewoo Kang. Self-attention graph pooling. In *ICML*, 2019.
- Jure Leskovec, Kevin J Lang, Anirban Dasgupta, and Michael W Mahoney. Statistical properties of community structure in large social and information networks. In *WWW*, 2008.
- Jiongqian Liang, Saket Gurukar, and Srinivasan Parthasarathy. MILE: A multi-level framework for scalable graph embedding. In *ICWSM*, 2021.
- Stuart Lloyd. Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 1982.
- Colin McDiarmid and Fiona Skerman. Modularity of erdős-rényi random graphs. *Random Structures & Algorithms*, 2020.
- Josiane Mothe, Karen Mkhitarian, and Mariam Haroutounian. Community detection: Comparison of state of the art algorithms. In *CSIT*. IEEE, 2017.
- Raj Rao Nadakuditi and Mark EJ Newman. Graph spectra and the detectability of community structure in networks. *Physical review letters*, 2012.
- Nicolò Navarin, Dinh V Tran, and Alessandro Sperduti. Pre-training graph neural networks with kernels. In *NeurIPS*, 2018.
- Mark EJ Newman. Modularity and community structure in networks. *PNAS*, 2006a.

- Mark EJ Newman. Finding community structure in networks using the eigenvectors of matrices. *Physical review E*, 2006b.
- Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *ICML*, 2016.
- Krzysztof Nowicki and Tom A B Snijders. Estimation and prediction for stochastic block-structures. *Journal of the American statistical association*, 96(455):1077–1087, 2001.
- Zizi Papacharissi. The virtual geographies of social networks: a comparative analysis of facebook, linkedin and asmallworld. *New media & society*, 2009.
- Tiago P Peixoto. Efficient monte carlo and greedy heuristic for the inference of stochastic block models. *Physical Review E*, 2014a.
- Tiago P. Peixoto. The graph-tool python library. *figshare*, 2014b. doi: 10.6084/m9.figshare.1164194. URL http://figshare.com/articles/graph_tool/1164194.
- Bryan Perozzi and Leman Akoglu. Scalable anomaly ranking of attributed neighborhoods. In *SDM*, 2016.
- Bryan Perozzi and Leman Akoglu. Discovering communities and anomalies in attributed graphs: Interactive visual exploration and summarization. *ACM TKDE*, 12(2), 2018.
- Bryan Perozzi, Leman Akoglu, Patricia Iglesias Sánchez, and Emmanuel Müller. Focused clustering and outlier detection in large attributed graphs. In *KDD*, 2014a.
- Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. DeepWalk: Online learning of social representations. In *KDD*, 2014b.
- Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 2008.
- Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 2008.
- Oleksandr Shchur and Stephan Günnemann. Overlapping community detection with graph neural networks. *arXiv preprint arXiv:1909.12201*, 2019.
- 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*, 2000.
- Tom AB Snijders and Krzysztof Nowicki. Estimation and prediction for stochastic block-models for graphs with latent block structure. *Journal of classification*, 1997.
- Jiaming Song and Stefano Ermon. Understanding the limitations of variational mutual information estimators. In *ICLR*, 2020.

Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: a simple way to prevent neural networks from overfitting. *JMLR*, 2014.

Fan-Yun Sun, Jordan Hoffmann, and Jian Tang. InfoGraph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *ICLR*, 2020.

Amanda L Traud, Eric D Kelsic, Peter J Mucha, and Mason A Porter. Comparing community structure to characteristics in online collegiate social networks. *SIAM review*, 2011.

Michael Tschannen, Josip Djolonga, Paul K Rubenstein, Sylvain Gelly, and Mario Lucic. On mutual information maximization for representation learning. In *ICLR*, 2020.

Anton Tsitsulin, Davide Mottin, Panagiotis Karras, and Emmanuel Müller. VERSE: Versatile graph embeddings from similarity measures. In *WWW*, 2018.

Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. Graph attention networks. In *ICLR*, 2017.

Petar Veličković, William Fedus, William L Hamilton, Pietro Liò, Yoshua Bengio, and R Devon Hjelm. Deep graph infomax. In *ICLR*, 2019.

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

Zhongdao Wang, Liang Zheng, Yali Li, and Shengjin Wang. Linkage based face clustering via graph convolution network. In *CVPR*, 2019b.

Yen-Chuen Wei and Chung-Kuan Cheng. Towards efficient hierarchical designs by ratio cut partitioning. In *IEEE International Conference on Computer-Aided Design*. IEEE, 1989.

Jaewon Yang and Jure Leskovec. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 2015.

Lei Yang, Xiaohang Zhan, Dapeng Chen, Junjie Yan, Chen Change Loy, and Dahua Lin. Learning to cluster faces on an affinity graph. In *CVPR*, 2019.

Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *KDD*, 2018a.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *NeurIPS*, 2018b.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. In *ICML*, 2019.

Xiaotong Zhang, Han Liu, Qimai Li, and Xiao-Ming Wu. Attributed graph clustering via adaptive graph convolution. In *IJCAI*, 2019.

Yunpeng Zhao, Elizaveta Levina, and Ji Zhu. Consistency of community detection in networks under degree-corrected stochastic block models. *The Annals of Statistics*, 2012.