Differentiable Community Detection with Graph Neural Networks and Stochastic Block Models

William Arliss
william.f.arliss@leidos.com
Leidos
Arlington, Virginia, USA

Graham Mueller
william.g.mueller@leidos.com
Leidos
Arlington, Virginia, USA

ABSTRACT

We propose a set of loss functions adapted from Stochastic Block Model (SBM) likelihood functions to train Graph Neural Networks (GNNs) for the task of unsupervised community detection. Identifying latent community structures is a prominent challenge for many graph applications. SBMs are classical models that describe the generating process of random graphs and are commonly used to infer community structure. The likelihood functions associated with SBMs are well-defined, differentiable, and measure the quality of inferred community partitions; this makes them particularly useful for unsupervised learning with GNNs. Our proposed loss functions are independent of any specific GNN architecture and demonstrate competitive or improved community detection performance against several alternatives. Evaluation is carried out with multiple architectures, offering a thorough empirical analysis of the state of community detection with GNNs.

CCS CONCEPTS

• **Computing methodologies** → **Neural networks**; **Cluster analysis**; • **Mathematics of computing** → **Random graphs**.

KEYWORDS

Stochastic Block Models, Graph Neural Networks, Unsupervised Learning, Community Detection

ACM Reference Format:
William Arliss and Graham Mueller. 2025. Differentiable Community Detection with Graph Neural Networks and Stochastic Block Models. In *Proceedings of Machine Learning on Graphs in the Era of Generative Artificial Intelligence at KDD (MLoG-GenAI '25).* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 INTRODUCTION

Graphs provide a rich source of relational information on which to perform a variety of machine learning tasks. Unsupervised community detection (also known as node clustering) refers to the problem of partitioning graph nodes into groups based on attributes and structural information. Methods for analyzing community structure are essential to applications in cybersecurity, social sciences, and e-commerce. For example, many Recommender Systems predict which product to recommend to a customer based on the inferred category (community) the product or customer belongs to.

A common tool for identifying and analyzing communities is the Stochastic Block Model (SBM) [26]. SBMs are statistical models of community structure in networks, parameterized by a node partition and a community structure matrix. Variations of the SBM have been proposed to address alternative assumptions about the generating processes of graphs [2, 9, 16, 20, 21, 27].

The progress of Graph Neural Networks (GNNs) in representation learning on graphs has motivated their use for community detection as well. Several GNN-based frameworks have been proposed [4, 29, 32, 39, 41] and demonstrate impressive performance for both semi-supervised and unsupervised community detection. Most unsupervised methods involve estimating the partition matrix through modularity maximization, link prediction, or solving the minimum-cut problem; typically a combination of custom GNN architectures and training routines are used.

Significant work has also been done to integrate the strong theoretical foundations of SBMs with the expressive power of GNNs [6, 10, 11, 24, 31]. Usually, these approaches incorporate GNNs as a component in a mixture model or as a Bayesian prior.

A natural synthesis of the two approaches is to incorporate an SBM likelihood function as a loss function for training a GNN [5]. Adapting the likelihood functions for different types of SBMs offers a set of configurable, fully differentiable objectives which can be used for unsupervised training of an arbitrary neural network.

This paper has two main contributions: (i) A set loss functions derived from SBM likelihood functions and (ii) an extensive comparison of unsupervised loss functions for the task of community detection. The loss functions are motivated by maximum likelihood estimation and the Graph Matching problem [33]. We compare the performance of GNN models trained with the SBM loss functions to several state-of-the-art alternatives on synthetic and real-world graphs.

The proposed loss functions are fully differentiable and do not require custom architectures or training routines. So for a fair empirical analysis, the same GNN architecture and training loop are used for each loss function. Consequently, comparison approaches that do not use stand-alone loss functions are not considered.

2 RELATED WORK

2.1 Stochastic Block Models

Stochastic Block Models, introduced in [26], are a family of generative models which assume that the existence of any edge in a graph is dependent only on the partition (community membership) of the two component nodes. SBMs are identified by a node partition

MLoG-GenAI '25, August, 2025, Toronto, ON

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

and a structure matrix, which defines the expectation of an edge between each partition.

Several variants of the SBM have been proposed to address different challenges: the Degree-Corrected SBM [16] mitigates the problem of skewed degree distributions by directly modeling degree heterogeneity; the Mixed Membership SBM [2] and the closely related Overlapping SBM [20] allow nodes to be members of more than one community, leading to more flexible interpretations of community structure; the Contextual SBM [9] incorporates node attributes, which are assumed to be generated conditionally on node communities; the Microcannonical SBM [27] enforces strict structural constraints in the model directly, rather than in expectation only. An in-depth review of these (and other) variants is given in [21].

The task of inference with SBMs typically involves identifying the process that generates a given graph and estimating the relevant parameters [21]. Inferring the partition of a graph from an SBM is sufficient for the task of community detection. A full survey of statistical community detection techniques related to SBM inference is given in [1].

The success of Graph Neural Networks in representation learning on graphs (see [13, 18, 34, 36]) has motivated several deep learning frameworks for SBMs. GNNs are used to parameterize Contextual SBMs in [6, 11, 31], where node features are assumed to be a function of community membership. Conversely, [10] uses a single-layer perceptron to model community membership as a function of node features. In [24] the authors design a variational auto-encoder GNN to parameterize an Overlapping SBM.

The SBM likelihood function is used as part of a composite loss function in [5]. In this work, the authors propose to combine an approximate SBM log-likelihood, a custom link prediction loss, an entropy term, and a task-specific loss in order to optimize a custom neural network framework. The framework is evaluated on the tasks of community detection, graph alignment, and anomalous correlation detection.

2.2 Deep Community Detection

Deep community detection refers to unsupervised or semi-supervised community detection performed with deep neural networks. Much work has been done (orthogonally to the SBM class) on GNN-based community detection.

In [39], a framework consisting of multiple GNN layers is proposed, where one module generates node embeddings and the other pools node features according to (predicted) community structure. A link prediction objective is used to guide the pooling function. In [4], the authors apply a GNN to the minimum-cut problem, which seeks to partition the set of nodes into disjoint (i.e., minimally connected) subsets by maximizing the average ratio of edges within communities to edges between communities. They do this by directly minimizing the negative of the minimum-cut metric plus a custom orthogonality constraint.

Both of the above approaches depend on custom architectures for task-specific problems. The focus of this paper, though, is on standalone objective functions that do not require custom architectures. One such general approach is taken in [29], where the authors propose using GNN embeddings to parameterize a Bernoulli-Poisson model [37, 42] of a graph. A likelihood-based loss function is derived from this model and edge sampling is used to address imbalance.

In [32] it is proposed to directly optimize modularity, a metric that measures the quality of community partitions. To train a GNN, the authors minimize the negative of estimated modularity plus a novel regularization term. The authors suggest that the orthogonality constraint from [4] tends to trap the optimization routine in local minima and instead devise their own "collapse regularization" meant to penalize trivial partitions. As a generalization of modularity optimization, [41] propose using the negative of the trace of the Markov Stability matrix [7, 8, 19] to train a GNN. Markov Stability is a dynamic quality metric that measures the probability that a random walk starting in one community will end in another after a certain number of time steps.

3 METHODS

3.1 Preliminary

Let $\mathcal{G} = (\mathcal{V}, \mathcal{E})$ be a directed simple graph with $n = |\mathcal{V}|$ nodes and $m = |\mathcal{E}|$ edges. Furthermore, let $\mathcal{D} = \mathcal{V} \times \mathcal{V}$ be the set of all possible node dyads (excluding self-loops) so that $\mathcal{E} \subseteq \mathcal{D}$. The $n \times n$ adjacency matrix $\mathbf{A}$ represents the edge structure of $\mathcal{G}$ and the n-vector $\mathbf{d}$ measures the node degrees such that $\mathbf{d}_u = \sum_{v=1}^{n} \mathbf{A}_{uv}$. Community memberships are represented in the k-partition matrix $\mathbf{Z} \in \{0,1\}^{n \times k}$, where $\mathbf{Z}_{ui} = 1$ if node $u \in \mathcal{V}$ is a member of community i and 0 otherwise.

3.2 Likelihood Functions

We now consider the likelihood functions of several Stochastic Block Model variants induced by different assumptions of the underlying generative process of $\mathcal{G}$. All models have parameters $\mathbf{Z}$ and $\mathbf{\Theta}$, where $\mathbf{\Theta}$ is a $k \times k$ structure matrix (also known as the block matrix). The block matrix is defined such that $\mathbf{\Theta}_{ij}$ is the expected number of edges from a node in community i to a node in community j.

3.2.1 Poisson. One common form of the SBM assumes that elements of $\mathbf{A}$ are Poisson distributed, conditional upon the community membership of the incident nodes [16, 21]. This gives the formal assumption $\mathbf{A}_{uv}|\mathbf{Z} \sim \text{Pois}(\mathbf{Z}'_u \mathbf{\Theta} \mathbf{Z}_v)$. Here, $\mathbf{\Theta}_{ij}$ is interpreted as the average number of edges between nodes in communities i and j. The likelihood of this model is

$$\mathcal{L}_{\text{P}}(\mathbf{Z}, \mathbf{\Theta}; \mathbf{A}) = \prod_{u \neq v} \frac{(\mathbf{Z}'_u \mathbf{\Theta} \mathbf{Z}_v)^{\mathbf{A}_{uv}}}{\mathbf{A}_{uv}!} \exp(-\mathbf{Z}'_u \mathbf{\Theta} \mathbf{Z}_v) \tag{1}$$

which is similar to the formulation in [16] except that self-loops are not considered here.

Recall that $\mathcal{G}$ is defined in this paper as a *simple* graph, which implies that $\mathbf{A}_{uv} \in \{0, 1\}$. Our analysis is thus restricted to the simple graph case, meaning that equation 1 is in fact a *partial* likelihood. The partial log-likelihood of this model is defined over the $\{0, 1\}$ sub-region of the standard Poisson support as

$$\ell_{\text{P}}(\mathbf{Z}, \mathbf{\Theta}; \mathbf{A}) = \sum_{u \neq v} \left[\mathbf{A}_{uv} \ln(\mathbf{Z}'_u \mathbf{\Theta} \mathbf{Z}_v) - \mathbf{Z}'_u \mathbf{\Theta} \mathbf{Z}_v \right]. \tag{2}$$

Note that the factorial term has been dropped because it is always equal to 0.

The maximum likelihood estimate (MLE) for $\boldsymbol{\Theta}$, derived in [16], is the ratio of the number of edges between two communities to the product of their community sizes. That is,

$$\hat{\boldsymbol{\Theta}}_{ij} = \left[\hat{\boldsymbol{\Theta}}(\mathcal{G}, \mathbf{Z})\right]_{ij} = \frac{\mathbf{M}_{ij}}{\mathbf{n}_i \mathbf{n}_j} \tag{3}$$

$$\mathbf{M} = \sum_{u,v \in \mathcal{E}} \mathbf{Z}_u \mathbf{Z}'_v \tag{4}$$

$$\mathbf{n} = \sum_{u=1}^{n} \mathbf{Z}_u. \tag{5}$$

Here, $\mathbf{M}$ is a $k \times k$ matrix such that $\mathbf{M}_{ij}$ is the number of edges from nodes in community i to nodes in community j and $\sum_{i=1}^{k}\sum_{j=1}^{k} \mathbf{M}_{ij} = m$. Also, $\mathbf{n}$ is a k-vector representing the number of nodes in each community and $\sum_{i=1}^{k} \mathbf{n}_i = n$.

3.2.2 Bernoulli. In focusing on simple graphs, it is useful to consider a model that fully aligns with the restriction on the adjacency matrix. An intuitive choice is to assume that elements in $\mathbf{A}$ are conditionally Bernoulli distributed [21]. Thus, the distribution assumption is modified to $\mathbf{A}_{uv}|\mathbf{Z} \sim \text{Bern}(\mathbf{Z}'_u \boldsymbol{\Theta} \mathbf{Z}_v)$. This new model interprets $\boldsymbol{\Theta}_{ij}$ as the probability of an edge between nodes in communities i and j. The log-likelihood of this model is

$$\begin{aligned}\ell_{\text{B}}(\mathbf{Z}, \boldsymbol{\Theta}; \mathbf{A}) = \sum_{u \neq v} \big[&\mathbf{A}_{uv} \ln(\mathbf{Z}'_u \boldsymbol{\Theta} \mathbf{Z}_v) \\ &+ (1 - \mathbf{A}_{uv}) \ln(1 - \mathbf{Z}'_u \boldsymbol{\Theta} \mathbf{Z}_v)\big].\end{aligned} \tag{6}$$

The advantage of this model over the partial Poisson model is that it is defined over the full range of support of the assumed distribution, not just a sub-region. The MLE of $\boldsymbol{\Theta}$ is the same as that of the Poisson model (equation 3).

3.2.3 Degree Correction. Another variety of the SBM, introduced in [16], seeks to incorporate degree heterogeneity into the model. The standard SBM assumes that each node has the same expected degree. This assumption can lead to a sub-optimal solutions on real-world networks, where degree distributions are observed to be highly skewed.

To address this, the n-vector $\boldsymbol{\phi}$ is introduced, allowing heterogeneous degree expectations. The expected value of edge (u, v) is now $\boldsymbol{\phi}_u \boldsymbol{\phi}_v \mathbf{Z}'_u \boldsymbol{\Theta} \mathbf{Z}_v$ instead of $\mathbf{Z}'_u \boldsymbol{\Theta} \mathbf{Z}_v$. The partial log-likelihood for the degree-corrected Poisson model is

$$\begin{aligned}\ell_{\text{P-DC}}(\mathbf{Z}, \boldsymbol{\Theta}, \boldsymbol{\phi}; \mathbf{A}) = \sum_{u \neq v} \big[&\mathbf{A}_{uv} \ln(\boldsymbol{\phi}_u \boldsymbol{\phi}_v \mathbf{Z}'_u \boldsymbol{\Theta} \mathbf{Z}_v) \\ &- \boldsymbol{\phi}_u \boldsymbol{\phi}_v \mathbf{Z}'_u \boldsymbol{\Theta} \mathbf{Z}_v\big]\end{aligned} \tag{7}$$

and the log-likelihood for the degree-corrected Bernoulli model, referred to as $\ell_{\text{B-DC}}$, is derived by the same substitution.

The MLE for $\boldsymbol{\phi}$, given in [16], is the ratio of a node's degree to the sum of degrees in that node's community. So the scaled degree correction of node u is

$$\hat{\boldsymbol{\phi}}_u = \left[\hat{\boldsymbol{\phi}}(\mathcal{G}, \mathbf{Z})\right]_u = (\mathbf{Z}'_u \mathbf{n}) \frac{\mathbf{d}_u}{\mathbf{Z}'_u \boldsymbol{\delta}} \tag{8}$$

$$\boldsymbol{\delta} = \sum_{u=1}^{n} \mathbf{Z}_u \mathbf{d}_u. \tag{9}$$

Here, $\boldsymbol{\delta}$ is a k-vector representing the sum of degrees in each community and $\sum_{i=1}^{k} \boldsymbol{\delta}_i = \sum_{u=1}^{n} \mathbf{d}_u$. Furthermore, $\mathbf{Z}'_u \boldsymbol{\delta}$ is the sum of degrees in the community that node u belongs to and $\mathbf{Z}'_u \mathbf{n}$ is the size of that community.

In the Bernoulli model, the unconstrained MLE is the same as that of the Poisson model (equation 8). However, the constraint that $0 < \boldsymbol{\phi}_u \boldsymbol{\phi}_v \mathbf{Z}'_u \boldsymbol{\Theta} \mathbf{Z}_v < 1$ for all (u, v) must be observed. If $\boldsymbol{\phi}_u \boldsymbol{\phi}_v$ scales the quantity to a value greater than 1, then the log-likelihood will be undefined. Therefore, it is necessary in $\ell_{\text{B-DC}}$ to impose the boundary $\boldsymbol{\phi}_u \leq 1$ for all u. We achieve this in practice by simply clamping the values to 1.

3.2.4 Overlap. The Overlapping SBM [20] assumes that a node can belong to more than one community. In this setting, the community of a given node varies depending on the edge it is observed in. To understand this model, first let $\mathbf{Z}^* \in \{0, 1\}^{n \times n \times k}$ be the expanded membership matrix, where $\mathbf{Z}^*_{u,v,i} = 1$ implies that node u is a member of the community i when it transmits an edge to node v. The expected value of edge (u, v) is therefore $\mathbf{Z}^{*\prime}_{uv} \boldsymbol{\Theta} \mathbf{Z}^*_{vu}$.

To reduce the dimensionality for the overlapping model, let $\mathbf{P} \in [0, 1]^{n \times k}$ be the collapsed membership matrix. This matrix is a summary of overlapping community memberships, defined as the degree-normalized sum over the second axis of the expanded membership matrix: $\mathbf{P}_u = \mathbf{d}_u^{-1} \sum_{v=1}^{n} \mathbf{Z}^*_{uv}$. In the overlapping setting, $\mathbf{P}_u$ is interpreted as the frequency of node u's membership in each of the k communities. Here, the expected value of edge (u, v) is $\mathbf{P}'_u \boldsymbol{\Theta} \mathbf{P}_v$. In the non-overlapping setting, the collapsed membership matrix $\mathbf{P}$ is equivalent to the partition matrix $\mathbf{Z}$. In both cases, $\sum_{i=1}^{k} \mathbf{P}_{ui} = 1$. The collapsed membership matrix will be relevant to neural network optimization.

3.3 Graph Neural Networks

With estimates of $\boldsymbol{\Theta}$ and $\boldsymbol{\phi}$ in place, we now turn to estimating the partition matrix. To begin, note that none of the log-likelihood functions described above are differentiable with respect to $\mathbf{Z}$, as it is a collection of discrete vectors. Because of this, gradient-based optimization methods are unavailable and Monte Carlo methods are typically used to find the likelihood-maximizing partition [1, 21]. To support the gradient-based optimization routine necessary to train a Graph Neural Network, some modifications must be made.

3.3.1 Parameter Specification. Neural networks are generally optimized with some variation of the gradient descent algorithm. For gradient descent to work in the SBM setting, an estimate of $\mathbf{Z}$ that is differentiable with respect to the neural network parameters is required.

We consider the collapsed membership matrix $\mathbf{P}$, which is a generalization of $\mathbf{Z}$, making it useful for both the standard and overlapping settings. The choice of $\mathbf{P}$ is convenient, as an estimate can be obtained by differentiable functions such as Softmax applied to GNN embeddings. Therefore, the SBM parameters of interest take the following forms:

$$\hat{\mathbf{P}} = \text{Softmax}(\text{GNN}(\mathcal{G}, \mathbf{X}; \mathbf{W})) \tag{10}$$

$$\hat{\boldsymbol{\Theta}} = \hat{\boldsymbol{\Theta}}(\mathcal{G}, \hat{\mathbf{P}}) \tag{11}$$

$$\hat{\boldsymbol{\phi}} = \hat{\boldsymbol{\phi}}(\mathcal{G}, \hat{\mathbf{P}}) \tag{12}$$

where GNN is any standard graph neural network with parameters $\mathbf{W}$. Here, $\hat{\mathbf{P}}$ is considered a relaxation of the partition matrix $\mathbf{Z}$ to a soft partition.

It should be noted that the output dimension of the GNN is the assumed number of communities k in the graph of interest. In practice, the exact number of communities need not be known a priori. Instead, the output dimension can be set to a reasonable overestimate and the model will learn the optimal number of communities (see Appendix D). Thus, k can be inferred as the number of unique elements in $\mathcal{K} = \{\arg\max \hat{\mathbf{P}}_u : u \in \mathcal{V}\}$. This is convenient for real-world graphs where the true number of communities may be unknown.

3.3.2 Loss Functions. The loss function associated with the Poisson model is the negative of the log-likelihood function in equation 2, with $\mathbf{Z}$ replaced by $\hat{\mathbf{P}}$ (equation 10) and $\boldsymbol{\Theta}$ replaced by $\hat{\boldsymbol{\Theta}}$ (equation 11). That is,

$$J_{\mathrm{P}}(\mathbf{W}) = -\sum_{u\neq v}\left[\mathbf{A}_{uv}\ln(\hat{\mathbf{P}}'_u\hat{\boldsymbol{\Theta}}\hat{\mathbf{P}}_v) - \hat{\mathbf{P}}'_u\hat{\boldsymbol{\Theta}}\hat{\mathbf{P}}_v\right]. \tag{13}$$

For brevity, the scalar $\hat{\pi}_{uv} = \hat{\mathbf{P}}'_u\hat{\boldsymbol{\Theta}}\hat{\mathbf{P}}_v$ is used for the remainder of this section. The loss function associated with the Bernoulli model is derived in the same way from equation 6, resulting in

$$J_{\mathrm{B}}(\mathbf{W}) = -\sum_{u\neq v}\left[\mathbf{A}_{uv}\ln(\hat{\pi}_{uv}) + (1-\mathbf{A}_{uv})\ln(1-\hat{\pi}_{uv})\right]. \tag{14}$$

Both losses are expressed as functions of the GNN parameters, $\mathbf{W}$.

Recall that $\mathbf{A}_{uv}$ is equal to 1 if $(u,v) \in \mathcal{E}$ and 0 if $(u,v) \notin \mathcal{E}$. Therefore, the loss function can be broken out into a summation over the positive edge set $\mathcal{E}$ and the negative edge set $\mathcal{N} = \mathcal{D} \setminus \mathcal{E}$. Doing so highlights the difference in cardinality of both sets. Often, real-world graphs can be highly sparse, meaning that $|\mathcal{N}| \gg |\mathcal{E}|$. Such imbalance can be problematic for optimizing a GNN. A common approach to address this is to under-sample [14] the majority class (usually the negative edge set) [13, 29, 38]. With this sampling approach, the loss functions are rewritten

$$J_{\mathrm{P}}(\mathbf{W}) = -\sum_{u,v\in\mathcal{E}}\left[\ln(\hat{\pi}_{uv}) - \hat{\pi}_{uv}\right] + \sum_{u,v\notin\mathcal{E}}\hat{\pi}_{uv} \tag{15}$$

$$\approx -\sum_{u,v\sim P_{\mathcal{E}}}\left[\ln(\hat{\pi}_{uv}) - \hat{\pi}_{uv}\right] + \eta^{-1}\sum_{u,v\sim P_{\mathcal{N}}}\hat{\pi}_{uv} \tag{16}$$

$$J_{\mathrm{B}}(\mathbf{W}) = -\sum_{u,v\in\mathcal{E}}\ln(\hat{\pi}_{uv}) - \sum_{u,v\notin\mathcal{E}}\ln(1-\hat{\pi}_{uv}) \tag{17}$$

$$\approx -\sum_{u,v\sim P_{\mathcal{E}}}\ln(\hat{\pi}_{uv}) - \eta^{-1}\sum_{u,v\sim P_{\mathcal{N}}}\ln(1-\hat{\pi}_{uv}). \tag{18}$$

The summations over $P_{\mathcal{E}}$ and $P_{\mathcal{N}}$ represent uniform samples from $\mathcal{E}$ and $\mathcal{N}$ with η as a scaling constant. For our experiments, all m positive edges are drawn deterministically and ηm samples from the negative set are drawn randomly at each training step.

The degree-corrected versions of both models are achieved by including $\hat{\boldsymbol{\phi}}$ (equation 12) in each loss function. Thus, the degree-corrected (DC) loss functions $J_{\text{P-DC}}$ and $J_{\text{B-DC}}$ are derived by substituting $\tilde{\pi}_{uv} = \hat{\boldsymbol{\phi}}_u\hat{\boldsymbol{\phi}}_v\hat{\mathbf{P}}'_u\hat{\boldsymbol{\Theta}}\hat{\mathbf{P}}_v$ for $\hat{\pi}_{uv}$ in the above equations.

3.3.3 Graph Matching. Another objective function is motivated by the Graph Matching problem [33, 35]. Graph Matching refers to the (approximate or exact) alignment of nodes across two graphs of possibly different sizes. Node alignment is usually defined by some real-world mechanism.

Let the block graph $\mathcal{G}_{\boldsymbol{\Theta}} = (\mathcal{V}_{\boldsymbol{\Theta}}, \mathcal{E}_{\boldsymbol{\Theta}})$ be defined with its (weighted) adjacency structure given by $\boldsymbol{\Theta}$. Each of its k nodes is a subset of nodes from $\mathcal{V}$; that is, node $i \in \mathcal{V}_{\boldsymbol{\Theta}}$ corresponds to the set $\{u \in \mathcal{V} : \mathbf{Z}_{ui} = 1\}$. Thus, $\mathbf{Z}$ is considered a mapping matrix which transforms the block matrix to the expectation of $\mathbf{A}$; that is, $\mathbf{Z}\boldsymbol{\Theta}\mathbf{Z}' \mapsto \mathrm{E}(\mathbf{A})$. Finding the optimal mapping matrix $\tilde{\mathbf{Z}}$ is the optimization problem

$$\underset{\tilde{\mathbf{Z}}\in\mathcal{Z}}{\operatorname{argmin}}\,\|\mathbf{A}-\tilde{\mathbf{Z}}\boldsymbol{\Theta}\tilde{\mathbf{Z}}'\|_F = \underset{\tilde{\mathbf{Z}}\in\mathcal{Z}}{\operatorname{argmin}}\,-\mathrm{tr}(\mathbf{A}'\tilde{\mathbf{Z}}\boldsymbol{\Theta}\tilde{\mathbf{Z}}') \tag{19}$$

where $\mathcal{Z} = \{\tilde{\mathbf{Z}} \in \{0,1\}^{n\times k} : \sum_{i=1}^{k}\tilde{\mathbf{Z}}_{ui} = 1 \text{ for all } u \in \mathcal{V}\}$ and $\mathrm{tr}(\cdot)$ is the matrix trace.

Because $\boldsymbol{\Theta}$ is an unknown parameter in the SBM formulation, we use its MLE $\hat{\boldsymbol{\Theta}}(\mathcal{G}, \tilde{\mathbf{Z}})$ from equation 3. Thus, we are seeking to find the (inverse) mapping of $\mathcal{G}$ to its estimated stochastic block representation $\mathcal{G}_{\hat{\boldsymbol{\Theta}}}$ by minimizing the quantity $-\mathrm{tr}(\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}')$. A proof of this statement is provided in Appendix B.

The optimization problem above is addressed with GNNs by substituting $\tilde{\mathbf{Z}}$ with the predicted membership matrix $\hat{\mathbf{P}}$ from equation 10 and using the block matrix estimate $\hat{\boldsymbol{\Theta}}$ from equation 11. The loss function in this setting is

$$J_{\mathrm{Match}}(\mathbf{W}) = -\mathrm{tr}(\mathbf{A}'\hat{\mathbf{P}}\hat{\boldsymbol{\Theta}}\hat{\mathbf{P}}') \tag{20}$$

and is again expressed as a function of the GNN parameters, $\mathbf{W}$. Note that $\mathbf{A}$ can be represented as a compressed sparse matrix, making the product $\mathbf{A}'\hat{\mathbf{P}}$ relatively efficient to compute. Another gain in computational efficiency comes from the relation

$$\mathrm{tr}(\mathbf{A}'\hat{\mathbf{P}}\hat{\boldsymbol{\Theta}}\hat{\mathbf{P}}') = \textstyle\sum_{u=1}^{n}\sum_{i=1}^{k}(\mathbf{A}'\hat{\mathbf{P}})_{ui}(\hat{\boldsymbol{\Theta}}\hat{\mathbf{P}}')_{iu}. \tag{21}$$

The use of sparse matrix multiplication results in a significant speedup compared to the edge sampling loss functions, as will be shown empirically in Section 4.

3.3.4 Regularization. The final component of the SBM loss framework is a regularization term. This regularization is meant to encourage the model to distribute the predicted partition across a sufficient number of communities and to ensure that the predicted partition is assortative [21] (i.e., edges occur mostly between nodes of the same community).

We propose a term that helps minimize the distance of the structure matrix diagonal from unity. The regularized form of an arbitrary loss function $J_\cdot$ is

$$J^*_\cdot(\mathbf{W}) = m^{-1}J_\cdot(\mathbf{W}) + \alpha\|\mathbf{1}_k - \hat{\boldsymbol{\theta}}_d\|_F \tag{22}$$

where $\hat{\boldsymbol{\theta}}_d$ is the diagonal of $\hat{\boldsymbol{\Theta}}$, $\mathbf{1}_k$ is a k-vector of ones, and $\|\cdot\|_F$ is the Frobenius norm. The hyper-parameter α controls the strength of the regularization.

This is analogous to the "collapse regularization" term introduced in [32] (see Appendix C.3). Both functions help to avoid trivial solutions to the optimization problem which arise when all nodes are assigned to one partition. Also, both are conveniently bounded in the interval $[0, \sqrt{k}]$.

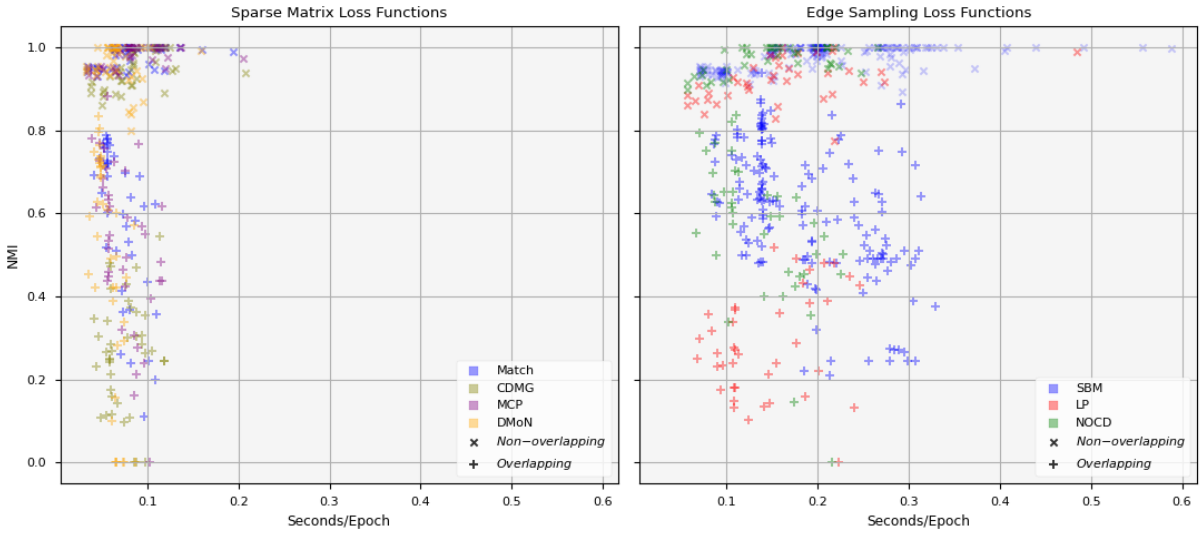

Figure 1: Accuracy (NMI) vs. training time (seconds-per-epoch) for each loss function on synthetic data. The left panel shows the loss functions that exploit matrix sparsity; the right panel shows loss functions that use edge sampling. The Bernoulli and Poisson SBM loss functions (and their degree-corrected variants) are all labeled "SBM" in the right panel. Results are marked by × for non-overlapping datasets and by + for overlapping datasets.

	SBM-4		SBM-8		SBM-16		SBM-32		*Avg.*	
	NMI	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*
J_{MCP}	99.2	99.2	99.3	99.0	98.6	95.8	94.5	**78.8**	97.9	93.2
J_{CDMG}	98.0	97.7	97.4	94.5	93.9	81.0	89.3	63.2	94.7	84.1
J_{DMoN}	89.3	87.6	98.7	98.1	**99.7**	**99.3**	94.5	78.1	95.5	90.8
J_{LP}	93.0	93.1	95.6	91.6	91.4	76.0	87.9	61.6	92.0	80.5
J_{NOCD}	97.5	97.9	99.2	98.7	98.9	96.4	92.3	71.5	97.0	91.1
J_{B}	**99.3**	**99.5**	99.3	98.5	98.7	95.2	94.1	75.1	97.9	92.1
J_{P}	98.3	98.3	**99.8**	**99.7**	99.2	97.0	93.7	74.8	97.7	92.4
$J_{\text{B-DC}}$	97.2	97.0	98.8	97.9	98.4	95.5	93.7	74.6	97.0	91.3
$J_{\text{P-DC}}$	98.5	98.6	99.4	99.2	99.7	99.1	94.3	75.9	**98.0**	**93.2**
J_{Match}	98.4	98.5	98.1	96.8	99.4	98.8	**94.7**	76.9	97.7	92.7

Table 1: Community detection performance on synthetic data with *non-overlapping* communities using GCN. Results are averaged over ten trials. The best scores (NMI and PF1) for each dataset are in bold.

	OSBM-4		OSBM-8		OSBM-16		OSBM-32		*Avg.*	
	NMI	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*
J_{MCP}	38.7	86.3	45.8	83.4	57.8	78.8	69.7	72.2	53.0	80.2
J_{CDMG}	27.2	69.5	26.2	57.1	28.3	57.1	23.5	55.3	26.3	59.7
J_{DMoN}	19.0	53.5	37.1	75.8	**74.6**	**90.6**	61.9	70.8	48.1	72.7
J_{LP}	34.7	57.6	29.3	68.2	23.7	62.2	26.2	52.7	28.5	60.2
J_{NOCD}	38.3	57.6	55.0	77.3	68.8	76.3	63.2	66.3	56.3	69.4
J_{B}	**56.8**	84.8	56.5	85.2	73.3	82.8	**70.5**	**73.7**	**64.3**	81.6
J_{P}	46.8	90.7	51.8	**89.7**	71.9	81.2	61.6	65.8	58.0	**81.8**
$J_{\text{B-DC}}$	50.8	93.4	**58.9**	84.8	73.8	77.5	67.2	69.4	62.7	81.3
$J_{\text{P-DC}}$	37.9	**97.2**	56.7	82.5	68.7	76.5	65.4	70.0	57.1	81.6
J_{Match}	36.0	85.0	54.2	75.4	72.2	86.4	69.3	73.4	57.9	80.0

Table 2: Community detection performance on synthetic data with *overlapping* communities using GCN. Results are averaged over ten trials. The best scores (overlapping NMI and PF1) for each dataset are in bold.

4 EXPERIMENTS

The proposed loss functions are evaluated based on the performance of the neural networks they are optimized with respect to. Community detection performance is measured by Normalized Mutual Information (NMI) and Pairwise-F1 (PF1) scores; where relevant, the overlapping variants [23] are used. Evaluation is carried out on a variety of synthetic and real-world datasets.

For baseline comparison, the loss functions used in several popular alternatives are also evaluated: Neural Overlapping Community Detection J_{NOCD} [29], Deep Modularity Network J_{DMoN} [32], Minimum-Cut Pooling J_{MCP} [4], Markov Stability J_{CDMG} [41], and link prediction J_{LP} [13]. These loss functions are described in greater detail in Appendix C. To ensure a fair comparison, a standard Graph Neural Network architecture is used for all experiments regardless of what was used in the original papers.

The architecture of choice is a two-layer Graph Convolutional Network (GCN) [18] with feature dropout and batch normalization. Additionally, we test the architecture described in [32], which we refer to as SkipGCN. We also provide aggregated results from several other architectures in Appendix E.

Implementation details are provided in Appendix A. The model configuration and hyperparameters were determined based on preliminary experimentation and are kept the same for all loss functions considered. An examination of (SBM-specific) hyperparameter sensitivity is provided in Appendix D.

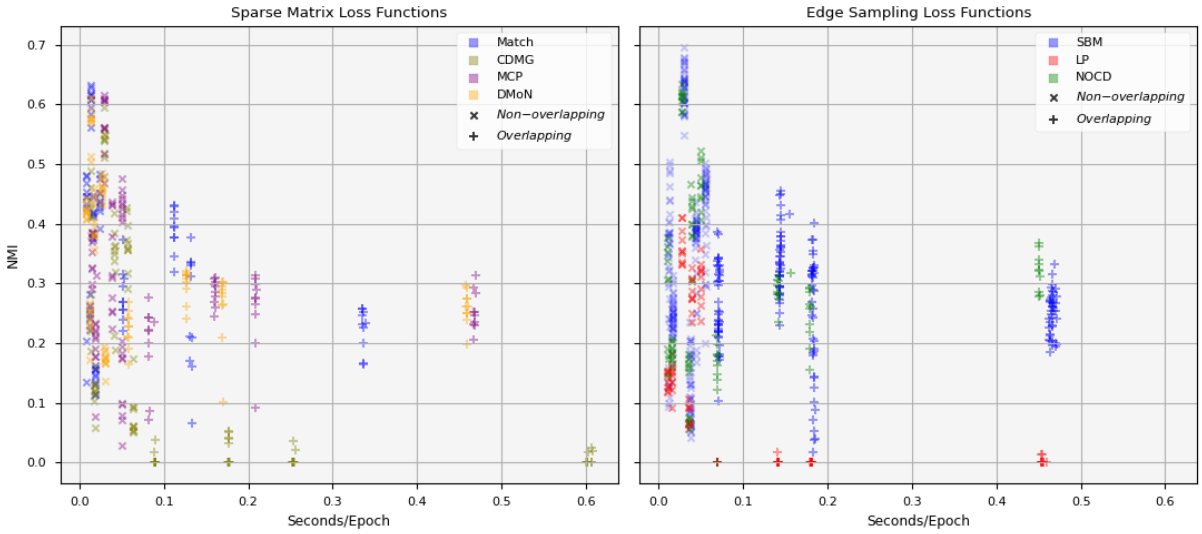

Figure 2: Accuracy (NMI) vs. training time (seconds-per-epoch) for each loss function on real-world data. The left panel shows the loss functions that exploit matrix sparsity; the right panel shows loss functions that use edge sampling. The Bernoulli and Poisson SBM loss functions (and their degree-corrected variants) are all labeled "SBM" in the right panel. Results are marked by × for non-overlapping datasets and by + for overlapping datasets.

	Cora		Citeseer		Pubmed		Wiki		ACB-Comp		ACB-Photo		*Avg.*	
	NMI	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*
J_{MCP}	34.2	22.6	19.4	16.4	13.0	19.5	33.1	26.7	43.4	45.8	57.1	52.6	33.4	30.6
J_{CDMG}	37.0	29.3	11.8	14.9	7.9	25.9	32.9	34.1	36.7	43.6	53.8	51.2	30.0	33.2
J_{DMoN}	**42.3**	25.9	23.9	19.1	**17.4**	20.4	**40.8**	36.0	45.9	34.6	57.0	47.1	**37.9**	30.5
J_{LP}	13.7	16.8	14.4	16.0	7.7	12.0	28.4	23.0	30.4	26.4	36.4	28.8	21.8	20.5
J_{NOCD}	27.3	19.8	18.6	19.8	12.7	19.1	40.6	34.6	**46.6**	41.7	61.1	56.9	34.5	32.0
J_{B}	29.9	27.9	24.8	**31.6**	12.4	33.1	36.5	35.8	42.8	**52.2**	64.2	60.8	35.1	40.2
J_{P}	28.2	25.8	22.8	30.0	13.5	**36.6**	36.2	36.5	40.2	51.3	**64.7**	**63.6**	34.3	**40.6**
$J_{\text{B-DC}}$	29.5	26.7	22.9	28.5	13.3	29.0	38.9	**38.4**	42.8	47.9	62.3	60.2	34.9	38.5
$J_{\text{P-DC}}$	41.3	**37.4**	**27.9**	28.2	10.8	28.3	33.6	31.1	45.8	32.6	60.1	53.4	36.6	35.2
J_{Match}	39.7	30.3	25.0	20.2	14.2	20.5	40.8	36.3	44.1	37.8	60.7	53.5	37.4	33.1

Table 3: Community detection performance on real-world data with *non-overlapping* communities using GCN. Results are averaged over ten trials. The best scores (NMI and PF1) for each dataset are in bold.

Nodes are split into a training, validation, and testing sets. Node embeddings are produced on the full training graph, then loss is computed with respect to subgraphs induced by the training node set. Early stopping and threshold identification are performed with respect to the validation nodes. The reported metrics are computed on the testing set and are averaged over ten trials.

4.1 Data

For experiments on synthetic data, graphs are generated according to the degree-corrected Poisson SBM (equation 1). The generated graphs are all assortative — that is, $\boldsymbol{\Theta}_{ii} > \boldsymbol{\Theta}_{ij}$ for all $i \neq j$ – with density between 0.01 and 0.06. Furthermore, degree distributions are sampled from a scale free distribution, where $\Pr(\mathbf{d}_u) \propto \mathbf{d}_u^{-\gamma}$ and $\gamma = 2.5$. Each graph has 2,000 nodes and 4, 8, 16, or 32 communities. The number of nodes in each community is normally distributed. For the overlapping setting, the partition and block matrices are randomly augmented.

To generate node features, length-100 vectors of means and variances are drawn from a multivariate normal distribution (squared for variances) for each community. Each node is given a feature vector sampled from a multivariate normal parameterized by the mean and variance of its assigned community. For overlapping communities, a node's feature vector is the average of the vectors sampled

	MAG-Chem		MAG-CS		MAG-Eng		MAG-Med		*Avg.*	
	NMI	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*
J_{MCP}	25.6	28.9	28.5	27.6	19.7	29.9	25.5	23.3	24.8	27.4
J_{CDMG}	0.6	34.6	2.2	28.7	0.6	**48.5**	0.6	23.4	1.0	33.8
J_{DMoN}	25.9	28.4	29.8	29.5	22.4	26.8	26.0	23.6	26.0	27.1
J_{LP}	0.0	9.7	0.2	10.6	0.0	10.0	0.3	17.1	0.1	11.8
J_{NOCD}	24.5	27.6	28.9	29.1	15.1	29.7	**32.0**	24.0	25.1	27.6
J_{B}	19.7	30.9	29.7	34.8	**29.4**	47.1	24.9	28.9	25.9	35.4
J_{P}	18.5	28.2	30.3	36.5	28.3	47.9	24.0	26.9	25.3	34.9
$J_{\text{B-DC}}$	26.4	32.8	35.7	44.2	25.7	48.3	26.9	**30.1**	28.7	**38.9**
$J_{\text{P-DC}}$	**30.8**	**39.4**	**40.5**	**44.9**	22.1	36.1	24.2	27.8	**29.4**	37.0
J_{Match}	25.0	32.9	38.9	42.1	28.0	37.4	22.3	25.3	28.6	34.4

Table 4: Community detection performance on real-world data with *overlapping* communities using GCN. Results are averaged over ten trials. The best scores (overlapping NMI and PF1) for each dataset are in bold.

for each of its communities. This method of attribute generation is similar to that described in [32].

For experiments on real-world data, ten common benchmark graphs are considered. Cora [22], Citeseer [12], and Pubmed [28] are all citation networks, where nodes are publications and edges indicate citations. Node attributes are vector representations of text associated with each publication and community partitions are the category of the publication. AmazonCoBuy (as presented in [30]) is a dataset of co-purchase graphs, where nodes are products and edges indicate products that are purchased together. Nodes are attributed by vector representations of user reviews and community partitions are the category of the products. The dataset is split into two graphs: computer products (ACB-Comp) and photo products (ACB-Photo). These graphs are accessed through DGL[1].

The Microsoft Academic Graph (as presented in [29]) is a dataset of co-authorship graphs, where nodes are authors and edges indicate co-authored publications. Node attributes are vectors of keywords associated with each author. Research areas form the overlapping community partition, as authors can research in multiple areas. The dataset is split into four graphs: chemistry (MAG-Chem), computer science (MAG-CS), engineering (MAG-Eng), and medicine (MAG-Med). These graphs are accessed through the GitHub repository[2] associated with [29].

Summary tables for the synthetic and real-world datasets are given in Appendix E.

4.2 Synthetic Results

We first evaluate community detection performance of each loss function on synthetic graphs without community overlap. Community predictions are the row-wise argmax of the model output and scoring is done with standard NMI and PF1. Results are shown in table 1. The top half of the table are comparison loss functions and the bottom half are our proposed SBM loss functions. The Bernoulli loss function performs best in terms of both NMI and PF1 on average. The MCP loss is also competitive.

[1]https://www.dgl.ai/dgl_docs/api/python/dgl.data.html
[2]https://github.com/shchur/overlapping-community-detection

For overlapping community detection, a threshold is applied to model outputs and the arguments exceeding that threshold are the predicted overlapping communities for each node. The threshold for each model is chosen to be accuracy maximizing. For evaluation, overlapping NMI and PF1 are used. Results are shown in table 2. The SBM loss functions generally outperform most comparison losses, with the DMoN loss function being competitive. On average, the Bernoulli variants perform best in overlapping NMI and PF1.

Figure 1 shows accuracy (measured as NMI) plotted against training time (measured in seconds-per-epoch) for each loss function. The left-hand panel shows the loss functions that exploit matrix sparsity: J_{Match}, J_{CDMG}, J_{MCP}, and J_{DMoN}. The right-hand panel shows the loss functions that employ (negative) edge sampling: J_{B}, J_{P}, J_{LP}, and J_{NOCD}. The Bernoulli and Poisson SBM loss functions (and their degree-corrected variants) have all been labeled "SBM" for simplicity in the right-hand panel. Note how the loss functions that exploit matrix sparsity are significantly faster than those that use edge sampling. The DMoN loss function is the fastest and our Graph Matching loss function is competitive in terms of both speed and accuracy.

4.3 Real-World Results

The community detection performance of each loss function is evaluated on real-world data in the same way as done on synthetic data. We first consider graphs without community overlap. Results are shown in table 3. The SBM loss functions outperform many of the comparison losses in terms of PF1 and NMI. The DMoN loss does best in average NMI and the Poisson loss does best in average PF1. The NOCD loss is also competitive.

Overlapping community detection performance is evaluated in the same way as the synthetic datasets. Results are shown in table 4. The SBM loss functions again outperform most comparison losses, with the degree-corrected variants doing best on average in terms of overlapping NMI and PF1.

Figure 2 again shows accuracy plotted against training time for each loss function. Notice how the gain in speed of the sparse matrix losses over the edge sampling losses is not as pronounced as in the synthetic datasets. This is likely due to the much higher dimension of node features in the real-world graphs.

To provide a better comparison with the DMoN approach from [32], we substitute SkipGCN layers for the standard GCN layers in our implementation. This introduces an additional weight matrix for nodes' own features instead of the self-loop augmentation on the adjacency matrix. Additionally, the SELU activation function is used instead of ReLU. While the other implementation details (e.g., hidden dimension, weight decay, etc.) are not identical, the component GNN layer is the same as that described in [32]. It should be noted that in [32], the authors compare their approach to several other approaches that are not GNN based, require custom training routines, or depend on very specific architectures (e.g., [4, 39]). They show that their DMoN approach is superior on a number of real-world graphs, including Cora, Citeseer, and Pubmed.

Table 5 shows the results of this comparison with the SkipGCN layer on non-overlapping graphs. The SBM loss functions generally outperform or are competitive with the DMoN loss function.

	Cora		Citeseer		Pubmed		Wiki		ACB-Comp		ACB-Photo		*Avg.*	
	NMI	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*
J_{MCP}	32.0	18.2	18.2	12.8	**19.1**	26.1	38.6	**37.8**	44.2	49.1	58.9	56.9	**35.1**	33.5
J_{CDMG}	38.7	29.5	12.7	13.2	6.1	15.6	26.2	29.4	27.8	36.2	44.5	46.5	26.0	28.4
J_{DMoN}	37.1	21.3	21.3	15.7	16.9	15.4	33.7	24.9	31.8	27.2	49.9	39.1	31.8	23.9
J_{LP}	13.2	11.2	6.9	8.4	5.0	9.4	21.8	16.2	23.1	16.2	25.4	19.8	15.9	13.5
J_{NOCD}	14.6	11.4	7.2	9.6	6.0	12.9	**38.6**	34.9	47.4	45.2	**63.2**	**62.9**	29.5	29.5
J_{B}	39.0	33.6	24.2	27.7	17.8	**30.7**	27.8	28.8	38.0	43.1	56.0	53.0	33.8	**36.1**
J_{P}	34.5	27.3	26.3	**30.2**	14.8	25.8	27.2	26.3	38.8	43.1	54.3	52.2	32.6	34.1
$J_{\text{B-DC}}$	36.6	31.2	24.6	25.4	13.6	25.5	25.3	27.1	33.8	38.9	53.6	51.2	31.2	33.2
$J_{\text{P-DC}}$	**43.3**	**36.0**	**26.5**	23.6	18.1	30.5	21.0	24.6	**47.8**	**53.5**	52.5	48.6	34.9	36.1
J_{Match}	41.7	29.0	25.8	23.7	14.2	18.0	29.8	23.2	39.5	33.7	54.9	49.7	34.3	29.5

Table 5: Community detection performance on real-world data *non-overlapping* communities using SkipGCN, as in [32]. Results are averaged over ten trials. The best scores (NMI and PF1) for each dataset are in bold.

Interestingly, the MCP and NOCD loss functions also perform well in this setting.

To further emphasize that this analysis is independent of specific GNN architectures, we provide aggregated results of additional experiments done with the following architectures in Appendix E: GraphSAGE [18], Graph Attention Network (GAT) [34], and Graph Isomorphism Network (GIN) [36].

4.4 Discussion

Figures 1 and 2 show how each of the loss functions compare to one-another in terms of both speed and accuracy. In general, the loss functions that exploit matrix sparsity are faster and just as accurate as those that use edge sampling. The difference in speed is less pronounced on the real-world graphs, which are larger and more sparse.

It should be noted that most losses are computed more quickly on non-overlapping datasets. This is simply because the (real-world) overlapping datasets under consideration happen to contain more nodes and edges than the non-overlapping datasets (see table 7 in Appendix E). In fact, every instance where "Seconds/Epoch" is greater than 0.25 corresponds to the MAG-Med dataset (with the exception of one occurrence of MAG-Chem), which is the graph with the greatest number of nodes and edges.

Another general trend is that the overlapping datasets usually produce lower accuracy scores than the non-overlapping ones. This limitation is not only observed for the SBM-based loss functions, but for the comparison losses as well. Further effort to adapt Overlapping SBMs to GNN loss functions is a direction for future work.

A key limitation of the proposed framework is that it is only designed for simple graphs. For multi-graphs or graphs with weighted edges, the Bernoulli and partial Poisson loss functions will not be applicable. The Poisson variant can easily be extended to support multi-graphs (see Appendix C.6), which allows the structure matrix to take values greater than 1. However, such a structure matrix does not conform with the regularization term in equation 22, which penalizes matrices with diagonals that are far from unity.

5 CONCLUSION

In this paper, a collection of loss functions are derived from Stochastic Block Model likelihood functions. These loss functions are configurable, fully differentiable, and theoretically grounded. They show strong performance in unsupervised training of Graph Neural Networks for community detection. An additional loss function is adapted from the Graph Matching problem and shows significant speed improvements. The proposed framework is subjected to extensive evaluation and shows competitive or improved performance against state-of-the-art loss functions.

ACKNOWLEDGMENTS

This research was conducted under the Graph Artificial Intelligence initiative of the Leidos Office of Technology. Export approval number: 25-LEIDOS-0428-29484. *This document does not contain export-controlled information as defined under the U.S. International Traffic in Arms Regulations or the U.S. Export Administration Regulations.*

REFERENCES

[1] Emmanuel Abbe. 2018. Community Detection and Stochastic Block Models: Recent Developments. *Journal of Machine Learning Research* 18, 177 (2018), 1–86. http://jmlr.org/papers/v18/16-480.html

[2] Edoardo M. Airoldi, David M. Blei, Stephen E. Fienberg, and Eric P. Xing. 2008. Mixed Membership Stochastic Blockmodels. *Journal of Machine Learning Research* 9, 65 (2008), 1981–2014. http://jmlr.org/papers/v9/airoldi08a.html

[3] Karol A. Bacik, Michael T. Schaub, Mariano Beguerisse-Díaz, Yazan N. Billeh, and Mauricio Barahona. 2016. Flow-Based Network Analysis of the Caenorhabditis elegans Connectome. *PLOS Computational Biology* 12, 8 (3 2016), e1005055. https://doi.org/10.1371/journal.pcbi.1005055

[4] Filippo Maria Bianchi, Daniele Grattarola, and Cesare Alippi. 2020. Spectral clustering with graph neural networks for graph pooling. In *Proceedings of the 37th International Conference on Machine Learning (ICML'20)*. JMLR.org, Article 82, 10 pages.

[5] Zheng Chen, Xinli Yu, Yuan Ling, and Xiaohua Hu. 2020. Neural Stochastic Block Model & Scalable Community-Based Graph Learning. arXiv:2005.07855 [cs.SI] https://arxiv.org/abs/2005.07855

[6] Eli Chien, Jianhao Peng, Pan Li, and Olgica Milenkovic. 2021. Adaptive Universal Generalized PageRank Graph Neural Network. arXiv:2006.07988 [cs.LG] https://arxiv.org/abs/2006.07988

[7] Jean-Charles Delvenne, Michael T. Schaub, Sophia N. Yaliraki, and Mauricio Barahona. 2013. *The Stability of a Graph Partition: A Dynamics-Based Framework for Community Detection.* Springer New York, 221–242. https://doi.org/10.1007/978-1-4614-6729-8_11

[8] J. C. Delvenne, S. N. Yaliraki, and M. Barahona. 2009. Stability of graph communities across time scales. arXiv:0812.1811 [physics.soc-ph] https://arxiv.org/abs/0812.1811
[9] Yash Deshpande, Andrea Montanari, Elchanan Mossel, and Subhabrata Sen. 2018. Contextual Stochastic Block Models. arXiv:1807.09596 [cs.SI] https://arxiv.org/abs/1807.09596
[10] O Duranthon and L Zdeborová. 2023. Neural-prior stochastic block model. *Machine Learning: Science and Technology* 4, 3 (Aug. 2023), 035017. https://doi.org/10.1088/2632-2153/ace60f
[11] Kimon Fountoulakis, Dake He, Silvio Lattanzi, Bryan Perozzi, Anton Tsitsulin, and Shenghao Yang. 2022. On Classification Thresholds for Graph Attention with Edge Features. arXiv:2210.10014 [cs.LG] https://arxiv.org/abs/2210.10014
[12] C. Lee Giles, Kurt D. Bollacker, and Steve Lawrence. 1998. CiteSeer: an automatic citation indexing system. In *Proceedings of the Third ACM Conference on Digital Libraries* (Pittsburgh, Pennsylvania, USA) *(DL '98)*. Association for Computing Machinery, New York, NY, USA, 89–98. https://doi.org/10.1145/276675.276685
[13] William L. Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems* (Long Beach, California, USA) *(NIPS'17)*. Curran Associates Inc., Red Hook, NY, USA, 1025–1035.
[14] Haibo He and Edwardo A. Garcia. 2009. Learning from Imbalanced Data. *IEEE Transactions on Knowledge and Data Engineering* 21, 9 (2009), 1263–1284. https://doi.org/10.1109/TKDE.2008.239
[15] Hidetaka Kamigaito and Katsuhiko Hayashi. 2021. Unified Interpretation of Softmax Cross-Entropy and Negative Sampling: With Case Study for Knowledge Graph Embedding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*. Association for Computational Linguistics. https://doi.org/10.18653/v1/2021.acl-long.429
[16] Brian Karrer and M. E. J. Newman. 2011. Stochastic blockmodels and community structure in networks. *Phys. Rev. E* 83 (1 2011), 016107. Issue 1. https://doi.org/10.1103/PhysRevE.83.016107
[17] Diederik P. Kingma and Jimmy Ba. 2017. Adam: A Method for Stochastic Optimization. arXiv:1412.6980 [cs.LG] https://arxiv.org/abs/1412.6980
[18] Thomas N. Kipf and Max Welling. 2017. Semi-Supervised Classification with Graph Convolutional Networks. arXiv:1609.02907 [cs.LG]
[19] Renaud Lambiotte, Jean-Charles Delvenne, and Mauricio Barahona. 2014. Random Walks, Markov Processes and the Multiscale Modular Organization of Complex Networks. *IEEE Transactions on Network Science and Engineering* 1, 2 (2014), 76–90. https://doi.org/10.1109/TNSE.2015.2391998
[20] Pierre Latouche, Etienne Birmelé, and Christophe Ambroise. 2011. Overlapping stochastic block models with application to the French political blogosphere. *The Annals of Applied Statistics* 5, 1 (March 2011). https://doi.org/10.1214/10-aoas382
[21] Clement Lee and Darren J. Wilkinson. 2019. A review of stochastic block models and extensions for graph clustering. *Applied Network Science* 4, 1 (Dec. 2019). https://doi.org/10.1007/s41109-019-0232-2
[22] Andrew McCallum, Kamal Nigam, Jason D. M. Rennie, and Kristie Seymore. 2000. Automating the Construction of Internet Portals with Machine Learning. *Information Retrieval* 3 (2000), 127–163. https://api.semanticscholar.org/CorpusID:349242
[23] Aaron F. McDaid, Derek Greene, and Neil Hurley. 2013. Normalized Mutual Information to evaluate overlapping community finding algorithms. arXiv:1110.2515 [physics.soc-ph] https://arxiv.org/abs/1110.2515
[24] Nikhil Mehta, Lawrence Carin Duke, and Piyush Rai. 2019. Stochastic Blockmodels meet Graph Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (Proceedings of Machine Learning Research, Vol. 97)*, Kamalika Chaudhuri and Ruslan Salakhutdinov (Eds.). PMLR, 4466–4474. https://proceedings.mlr.press/v97/mehta19a.html
[25] M. E. J. Newman and M. Girvan. 2004. Finding and evaluating community structure in networks. *Physical Review E* 69, 2 (Feb. 2004). https://doi.org/10.1103/physreve.69.026113
[26] Krzysztof Nowicki and Tom A. B. Snijders. 2001. Estimation and Prediction for Stochastic Blockstructures. *J. Amer. Statist. Assoc.* 96 (2001), 1077 – 1087. https://api.semanticscholar.org/CorpusID:9478789
[27] Tiago P. Peixoto. 2017. Nonparametric Bayesian inference of the microcanonical stochastic block model. *Physical Review E* 95, 1 (Jan. 2017). https://doi.org/10.1103/physreve.95.012317
[28] Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Gallagher, and Tina Eliassi-Rad. 2008. Collective Classification in Network Data. In *The AI Magazine*. https://api.semanticscholar.org/CorpusID:62016134
[29] Oleksandr Shchur and Stephan Günnemann. 2019. Overlapping Community Detection with Graph Neural Networks. *Deep Learning on Graphs Workshop, KDD* abs/1909.12201 (2019). https://api.semanticscholar.org/CorpusID:88492570
[30] Oleksandr Shchur, Maximilian Mumme, Aleksandar Bojchevski, and Stephan Günnemann. 2018. Pitfalls of Graph Neural Network Evaluation. *Relational Representation Learning Workshop, NeurIPS 2018* (2018).
[31] Cheng Shi, Liming Pan, Hong Hu, and Ivan Dokmanić. 2024. Homophily modulates double descent generalization in graph convolution networks. arXiv:2212.13069 [cs.LG] https://arxiv.org/abs/2212.13069
[32] Anton Tsitsulin, John Palowitch, Bryan Perozzi, and Emmanuel Müller. 2024. Graph clustering with graph neural networks. *J. Mach. Learn. Res.* 24, 1, Article 127 (3 2024), 21 pages.
[33] S. Umeyama. 1988. An eigendecomposition approach to weighted graph matching problems. *IEEE Transactions on Pattern Analysis and Machine Intelligence* 10, 5 (1988), 695–703. https://doi.org/10.1109/34.6778
[34] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Liò, and Yoshua Bengio. 2018. Graph Attention Networks. arXiv:1710.10903 [stat.ML]
[35] Joshua T Vogelstein, John M Conroy, Vince Lyzinski, Louis J Podrazik, Steven G Kratzer, Eric T Harley, Donniell E Fishkind, R Jacob Vogelstein, and Carey E Priebe. 2015. Fast approximate quadratic programming for graph matching. *PLOS one* 10, 4 (2015), e0121002.
[36] Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. 2019. How Powerful are Graph Neural Networks? arXiv:1810.00826 [cs.LG]
[37] Jaewon Yang and Jure Leskovec. 2013. Overlapping community detection at scale: a nonnegative matrix factorization approach. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining* (Rome, Italy) *(WSDM '13)*. Association for Computing Machinery, New York, NY, USA, 587–596. https://doi.org/10.1145/2433396.2433471
[38] Zhen Yang, Ming Ding, Chang Zhou, Hongxia Yang, Jingren Zhou, and Jie Tang. 2020. Understanding Negative Sampling in Graph Representation Learning. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining* (Virtual Event, CA, USA) *(KDD '20)*. Association for Computing Machinery, New York, NY, USA, 1666–1676. https://doi.org/10.1145/3394486.3403218
[39] Rex Ying, Jiaxuan You, Christopher Morris, Xiang Ren, William L. Hamilton, and Jure Leskovec. 2018. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd International Conference on Neural Information Processing Systems* (Montréal, Canada) *(NIPS'18)*. Curran Associates Inc., Red Hook, NY, USA, 4805–4815.
[40] Yu and Shi. 2003. Multiclass spectral clustering. In *Proceedings Ninth IEEE International Conference on Computer Vision*. 313–319 vol.1. https://doi.org/10.1109/ICCV.2003.1238361
[41] Shunjie Yuan, Chao Wang, Qi Jiang, and Jianfeng Ma. 2022. Community Detection with Graph Neural Network using Markov Stability. In *2022 International Conference on Artificial Intelligence in Information and Communication (ICAIIC)*. 437–442. https://doi.org/10.1109/ICAIIC54071.2022.9722614
[42] Mingyuan Zhou. 2015. Infinite Edge Partition Models for Overlapping Community Detection and Link Prediction. In *Proceedings of the Eighteenth International Conference on Artificial Intelligence and Statistics (Proceedings of Machine Learning Research, Vol. 38)*, Guy Lebanon and S. V. N. Vishwanathan (Eds.). PMLR, San Diego, California, USA, 1135–1143. https://proceedings.mlr.press/v38/zhou15a.html

A IMPLEMENTATION

The neural network used for all experiments consists of two GNN layers with feature dropout, batch normalization, and ReLU activation in-between. The default GNN layer is GCN [18] unless otherwise specified. The dimension of the output layer is 25 for all experiments and the activation is determined by the loss function. The hidden dimension is 250 for synthetic graphs and 500 for real-world graphs. The dropout rate is 0.5. Weight decay is applied with a strength of 0.0001. Adam optimization [17] is used with a learning rate of 0.0001. Training is carried out for a maximum of 500 epochs; early stopping is evaluated every 5 epochs and is engaged after 10 evaluations with no improvement. The SBM loss functions all take the form of equation 22 with regularization strength $\alpha = 1.0$. The collapse regularization strength for the DMoN loss is the same. For loss functions that support negative sampling, 3 negative edges are drawn for every positive edges, and balanced weighting is applied. The train-val-test split is 60-20-20.

Experiments are done in Python 3.8 with DGL[3] and PyTorch[4]. Synthetic experiments are conducted with a 2.4 GHz Intel Core i9

[3] https://www.dgl.ai/
[4] https://pytorch.org/

processor and 32 GB of memory. Real-world experiments are conducted with a 2.5 GHz Intel(R) Xeon(R) Platinum 8259CL processor and 15 GB of memory and a Tesla T4 GPU and Cuda version 12.4.

B GRAPH MATCHING PROOF

Let $\mathcal{Z} = \{\tilde{\mathbf{Z}} \in \{0,1\}^{n\times k} : \sum_{i=1}^{k} \tilde{\mathbf{Z}}_{ui} = 1 \text{ for all } u \in \mathcal{V}\}$. Consider the distance quantity

$$\begin{aligned} q &= \left\|\mathbf{A} - \tilde{\mathbf{Z}}\boldsymbol{\Theta}\tilde{\mathbf{Z}}'\right\|_F^2 \\ &= \|\mathbf{A}\|_F^2 + \left\|\tilde{\mathbf{Z}}\boldsymbol{\Theta}\tilde{\mathbf{Z}}'\right\|_F^2 - 2\,\mathrm{tr}\left(\mathbf{A}'\tilde{\mathbf{Z}}\boldsymbol{\Theta}\tilde{\mathbf{Z}}'\right). \end{aligned}$$

Minimizing q with respect to $\boldsymbol{\Theta}$ gives

$$\begin{aligned} \frac{\partial q}{\partial \boldsymbol{\Theta}} &= 2\tilde{\mathbf{Z}}\boldsymbol{\Theta}'\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}' - 2\mathbf{A}'\tilde{\mathbf{Z}}\tilde{\mathbf{Z}}' \overset{\text{set}}{=} 0 \\ &\implies \tilde{\boldsymbol{\Theta}} = (\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}\tilde{\mathbf{Z}}'\mathbf{A}\tilde{\mathbf{Z}}(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}. \end{aligned}$$

Now recall the MLE $\hat{\boldsymbol{\Theta}}$ from equation 3:

$$\begin{aligned} \hat{\boldsymbol{\Theta}}(\mathcal{G}, \tilde{\mathbf{Z}}) &= \mathbf{M} \oslash \mathbf{n}\mathbf{n}' \\ \mathbf{M} &= \sum_{u,v\in\mathcal{E}} \tilde{\mathbf{Z}}_u\tilde{\mathbf{Z}}_v' = \tilde{\mathbf{Z}}'\mathbf{A}\tilde{\mathbf{Z}} \\ \mathbf{n} &= \sum_{u=1}^{n} \tilde{\mathbf{Z}}_u \end{aligned}$$

where $\oslash$ represents element-wise division. When $\tilde{\mathbf{Z}} \in \mathcal{Z}$, note that $\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}} = \mathrm{diag}(\mathbf{n})$ and $(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1} = \mathrm{diag}(\mathbf{n}^{-1})$ where $\mathbf{n}^{-1}$ is the element-wise inverse of $\mathbf{n}$. Furthermore, $[(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}]_{ii} = 1/(\mathbf{n}\mathbf{n}')_{ii}$ for all $i = 1, ..., k$. It can also be verified that

$$(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}\mathbf{M}(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1} = \mathbf{M} \oslash \mathbf{n}\mathbf{n}'.$$

Therefore, the distance minimizer $\tilde{\boldsymbol{\Theta}}$ is equivalent to the maximum likelihood estimate $\hat{\boldsymbol{\Theta}}$ in this case.

We can now substitute the MLE into q as done in Section 3.3.3:

$$\begin{aligned} q &= \|\mathbf{A}\|_F^2 + \left\|\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}'\right\|_F^2 - 2\,\mathrm{tr}\left(\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}'\right) \\ &= \|\mathbf{A}\|_F^2 + \left\|\tilde{\mathbf{Z}}(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}\tilde{\mathbf{Z}}'\mathbf{A}\tilde{\mathbf{Z}}(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}\tilde{\mathbf{Z}}'\right\|_F^2 - 2\,\mathrm{tr}\left(\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}'\right) \\ &= \|\mathbf{A}\|_F^2 + \mathrm{tr}\left(\tilde{\mathbf{Z}}(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}\tilde{\mathbf{Z}}'\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}'\right) - 2\,\mathrm{tr}\left(\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}'\right) \\ &= \|\mathbf{A}\|_F^2 - \mathrm{tr}\left(\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}'\right). \end{aligned}$$

The final equality comes from the cyclic property of the matrix trace:

$$\begin{aligned} \mathrm{tr}\left(\tilde{\mathbf{Z}}(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}\tilde{\mathbf{Z}}'\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}'\right) &= \mathrm{tr}(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}}(\tilde{\mathbf{Z}}'\tilde{\mathbf{Z}})^{-1}\tilde{\mathbf{Z}}'\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}) \\ &= \mathrm{tr}\left(\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}'\right) \end{aligned}$$

Dropping the constant term, we have the result

$$\underset{\tilde{\mathbf{Z}}\in\mathcal{Z}}{\mathrm{argmin}} \|\mathbf{A} - \tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}'\|_F^2 = \underset{\tilde{\mathbf{Z}}\in\mathcal{Z}}{\mathrm{argmin}} \; -\mathrm{tr}(\mathbf{A}'\tilde{\mathbf{Z}}\hat{\boldsymbol{\Theta}}\tilde{\mathbf{Z}}').$$

which (noting that $\|\cdot\|_F^2$ is a monotonic transformation of $\|\cdot\|_F$ in equation 19) proves the statement in Section 3.3.3.

Substituting predicted membership $\hat{\mathbf{P}}$ for the mapping matrix $\tilde{\mathbf{Z}}$ is a useful approximation of this result when training a GNN.

C ADDITIONAL OBJECTIVE FUNCTIONS

C.1 Bernoulli-Poisson

The Bernoulli-Poisson model, explored in [29, 37], assumes $\mathbf{A}_{uv}|\mathbf{Z} \sim \mathrm{Bern}(1-e^{-\mathbf{Z}_u'\mathbf{Z}_v})$. In [29], the "Neural Overlapping Community Detection" (NOCD) model is introduced, which derives a loss function from the Bernoulli-Poisson likelihood function. The authors propose

$$\begin{aligned} \hat{\mathbf{Z}} &= \mathrm{ReLU}(\mathrm{GNN}(\mathcal{G}, \mathbf{X}; \mathbf{W})) \\ J_{\mathrm{NOCD}}(\mathbf{W}) &= -\sum_{u,v\sim P_{\mathcal{E}}} \ln(1-\exp(-\hat{\mathbf{Z}}_u'\hat{\mathbf{Z}}_v)) \\ &\quad + \eta^{-1}\sum_{u,v\sim P_{\mathcal{N}}} \hat{\mathbf{Z}}_u'\hat{\mathbf{Z}}_v. \end{aligned} \tag{23}$$

This is similar to the Poisson SBM (equation 16), with one difference being that there is no block matrix. Also, a complete derivation of the Poisson log-likelihood is used instead of a partial Poisson log-likelihood, as in equation 2.

C.2 Link Prediction

A common objective function used for graph representation learning is cross-entropy-based link prediction loss. This model seeks to estimate the adjacency structure of a graph and is motivated by the assumption $\mathbf{A}_{uv}|\mathbf{Z} \sim \mathrm{Bern}(\sigma(\mathbf{z}_u'\mathbf{z}_v))$. One popular formulation [13, 15, 38] is

$$\begin{aligned} \hat{\mathbf{Z}} &= \mathrm{GNN}(\mathcal{G}, \mathbf{X}; \mathbf{W}) \\ J_{\mathrm{LP}}(\mathbf{W}) &= -\sum_{u,v\sim P_{\mathcal{E}}} \ln\left(\sigma(\hat{\mathbf{Z}}_u'\hat{\mathbf{Z}}_v)\right) \\ &\quad - \eta^{-1}\sum_{u,v\sim P_{\mathcal{N}}} \ln\left(1-\sigma(\hat{\mathbf{Z}}_u'\hat{\mathbf{Z}}_v)\right) \end{aligned} \tag{24}$$

where σ is the sigmoid function. This can be viewed as an analog to the Bernoulli SBM (equation 18) with the key difference being the absence of a block matrix.

C.3 Modularity

In [32], it is proposed to directly maximize a graph partition quality metric. The approach, referred to as "Deep Modularity Network" (DMoN), is focused on modularity [25], defined as

$$Q = \frac{1}{2m}\mathrm{tr}\left(\mathbf{Z}'\left(\mathbf{A} - \frac{\mathbf{d}\mathbf{d}'}{2m}\right)\mathbf{Z}\right). \tag{25}$$

The objective is to maximize modularity, or equivalently minimize the negative of modularity. Thus, the loss function is

$$\begin{aligned} \hat{\mathbf{Z}} &= \mathrm{Softmax}(\mathrm{GNN}(\mathcal{G}, \mathbf{X}; \mathbf{W})) \\ J_{\mathrm{DMoN}}(\mathbf{W}) &= -\frac{1}{2m}\,\mathrm{tr}\left(\hat{\mathbf{Z}}'\left(\mathbf{A} - \frac{\mathbf{d}\mathbf{d}'}{2m}\right)\hat{\mathbf{Z}}\right) \\ &\quad + \alpha\left(\frac{\sqrt{k}}{n}\left\|\sum_{u}^{n}\hat{\mathbf{Z}}_u'\right\|_F - 1\right) \end{aligned} \tag{26}$$

where the first term minimizes negative modularity and the second term is a "collapse regularization" meant to prevent all nodes from being assigned to the same block (with regularization strength given by hyperparameter α) [32].

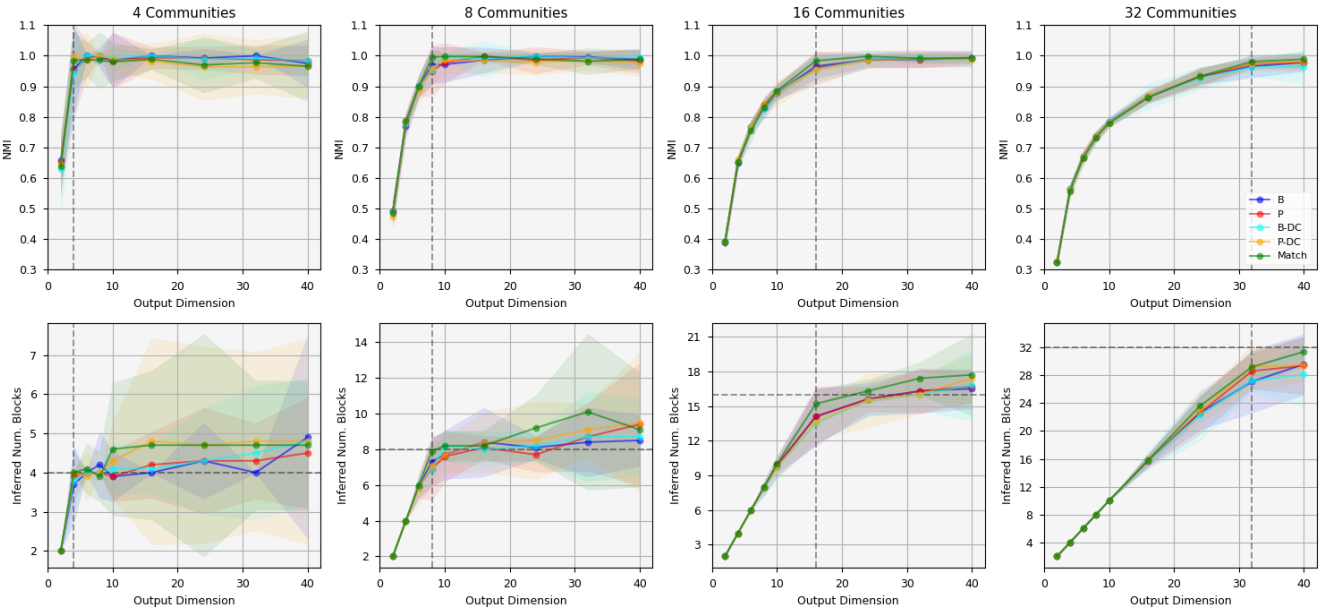

Figure 3: Top row: Accuracy (NMI) vs. GNN output dimension for different numbers of communities. Bottom row: Inferred number of communities vs. GNN output dimension. Columns: Actual number of communities in the graph.

C.4 Markov Stability

Another perspective of modularity is taken in [41]. The approach, referred to as "Community Detection based on Markov Stability and Graph Neural Network" (CDMG), seeks to maximize a dynamic graph partition quality metric. Markov Stability [7, 8, 19] is defined

$$R_t = \operatorname{tr}\left(\mathbf{Z}'\left(\mathbf{\Pi}\mathbf{M}^t - \boldsymbol{\pi}'\boldsymbol{\pi}\right)\mathbf{Z}\right) \tag{27}$$

where $\boldsymbol{\pi} = \mathbf{d}'(2m)^{-1}$, $\mathbf{\Pi} = \operatorname{diag}(\boldsymbol{\pi})$, $\mathbf{M} = \mathbf{D}^{-1}\mathbf{A}$, and $\mathbf{D} = \operatorname{diag}(\mathbf{d})$. The matrix in this expression represents the probability that a random walk starting in one community will end in another after a certain number of time steps. This is also equivalent to the modularity in equation 25 when $t = 1$ [19].

The CDMG approach seeks to maximize Markov Stability by minimizing its negative:

$$\hat{\mathbf{Z}} = \operatorname{ReLU}(\operatorname{GNN}(\mathcal{G}, \mathbf{X}; \mathbf{W}))$$
$$J_{\text{CDMG}}(\mathbf{W}) = -\operatorname{tr}\left(\hat{\mathbf{Z}}'\left(\mathbf{\Pi}\mathbf{M}^t - \boldsymbol{\pi}'\boldsymbol{\pi}\right)\hat{\mathbf{Z}}\right) \tag{28}$$

where t is a hyperparameter that can be tuned according to the resolution of the graph communities. Larger values of t detect coarser communities [3, 7, 8, 19, 41]. We use $t = 1$ for all experiments.

C.5 Minimum-Cut

An objective function motivated by the minimum-cut problem is proposed in [4]. The minimum-cut problem is a task that seeks to find the partition that minimizes the number of edges between groups. This is done by maximizing the ratio of the number of edges within a group to the number of edges between groups in the rest of the graph. Formally,

$$\max \frac{1}{k}\sum_{i=1}^{k}\frac{\mathbf{Z}'_{\cdot i}\mathbf{A}\mathbf{Z}_{\cdot i}}{\mathbf{Z}'_{\cdot i}\mathbf{D}\mathbf{Z}_{\cdot i}} \tag{29}$$

where $\mathbf{Z}_{\cdot i} \in \{0,1\}^n$ is the i^{th} column of $\mathbf{Z}$ [40].

The proposed approach, known as MinCutPool [4], uses a pooling-based architecture with the softmax function applied to outputs to estimate community assignments. The objective function used for training is

$$\hat{\mathbf{Z}} = \operatorname{Softmax}(\operatorname{GNN}(\mathcal{G}, \mathbf{X}; \mathbf{W}))$$
$$J_{\text{MCP}}(\mathbf{W}) = -\frac{\operatorname{tr}(\hat{\mathbf{Z}}'\tilde{\mathbf{A}}\hat{\mathbf{Z}})}{\operatorname{tr}(\hat{\mathbf{Z}}'\tilde{\mathbf{D}}\hat{\mathbf{Z}})} + \left\| \frac{\hat{\mathbf{Z}}'\hat{\mathbf{Z}}}{\left\|\hat{\mathbf{Z}}'\hat{\mathbf{Z}}\right\|_F} - \frac{\mathbf{I}_k}{\sqrt{k}} \right\|_F \tag{30}$$

where $\mathbf{I}_k$ is the $k \times k$ identity matrix, $\tilde{\mathbf{A}} = \mathbf{D}^{-1/2}\mathbf{A}\mathbf{D}^{1/2}$ is the normalized adjacency matrix, and $\tilde{\mathbf{D}} = \operatorname{diag}(\tilde{\mathbf{A}}\mathbf{1}_n)$ is the normalized degree matrix. The second term is an orthogonality penalty, meant to encourage orthogonality between communities and uniformity in community sizes.

C.6 Poisson

The partial Poisson loss function in section 3.3.2 can be extended to support multi-graphs, where $\mathbf{A}_{uv} \in \mathbb{N}_0$. To do so, the same log-likelihood function (equation 2) is considered. Note that the factorial term from equation 1 is dropped because its derivative is zero with respect to the parameters. Consequently, equation 13 is a valid loss function for both simple graphs and multi-graphs. In order to incorporate negative sampling, equation 16 is modified to

$$J_{\text{P}^*}(\mathbf{W}) \approx -\sum_{u,v\sim P_{\mathcal{E}}}\left[\mathbf{A}_{uv}\ln(\hat{\pi}_{uv}) - \hat{\pi}_{uv}\right] + \eta^{-1}\sum_{u,v\sim P_N}\hat{\pi}_{uv} \tag{31}$$

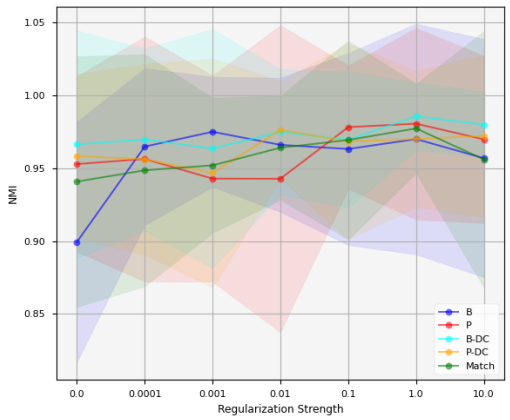

Figure 4: Accuracy (NMI) vs. regularization strength. Horizontal axis is not to scale.

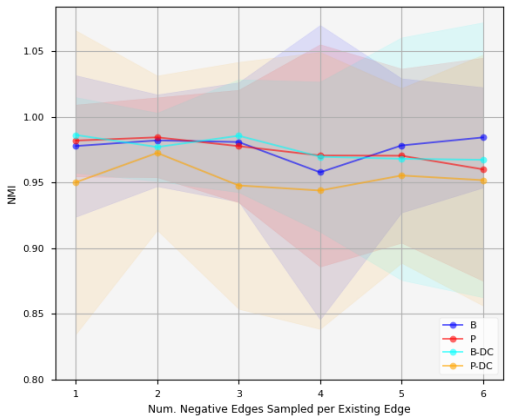

Figure 5: Accuracy (NMI) vs. number of negative edges sampled per each existing edge.

where η is scaling constant meant to balance the contribution of the positive and negative edge sets to the total loss. Note that J_{P^*} is identical to J_{P} for simple graphs.

D HYPERPARAMETER SENSITIVITY

This section provides an examination of SBM-specific hyperparameters. For these experiments, consideration is restricted to the non-overlapping setting and synthetically generated graphs are used.

To begin, we consider the relation of the output dimension d of the GNN embeddings to the number of effective clusters found in a given graph. The top row of figure 3 displays accuracy (measured as NMI) plotted against output dimension. The bottom row shows the inferred number of clusters (computed as $|\mathcal{K}|$) against output dimension. Results are averaged over ten trials for graphs with $k = 4, 8, 16, 32$ actual clusters. When the output dimension is smaller than the true number of clusters, there is generally poor accuracy and $|\mathcal{K}| = d$. When the output dimension is greater than the true number of clusters, accuracy is much higher and $|\mathcal{K}| \approx k$. The variance of community detection performance (in terms of accuracy and ability to recover the true number of clusters) tends to be greater when the output dimension is slightly larger than k. It is also worth noting that J_{Match} tends to overestimate the number of clusters.

Next, we look the impact of the regularization strength parameter α from equation 22. Figure 4 shows accuracy measured against different values of α averaged over ten trials. Note that the horizontal axis is not to scale. There is a moderate upward trend, suggesting that the regularization term does contribute to community detection performance.

Finally, the effect of negative sampling is studied. Figure 5 plots accuracy measured against different numbers of negative edges sampled per each existing edge (referred to as η). Accuracy appears fairly consistent across different levels of negative sampling, with a slight drop-off after $\eta = 3$. For this experiment, the loss function is weighted such that negative and positive edges have equal importance.

E ADDITIONAL TABLES

Name	Nodes	Edges	Partitions	Dimension	Overlap
SBM-4	2,000	202,102	4	100	No
SBM-8	2,000	150,648	8	100	No
SBM-16	2,000	96,436	16	100	No
SBM-32	2,000	56,526	32	100	No
OSBM-4	2,000	240,242	4	100	Yes
OSBM-8	2,000	196,510	8	100	Yes
OSBM-16	2,000	132,976	16	100	Yes
OSBM-32	2,000	72,422	32	100	Yes

Table 6: Synthetic graph summaries.

Name	Nodes	Edges	Partitions	Dimension	Overlap
Cora	2,708	10,556	7	1,433	No
Citeseer	3,327	9,228	6	3,703	No
Pubmed	19,717	88,651	3	500	No
Wiki	11,367	431,726	10	300	No
ACB-Comp	13,752	491,722	10	767	No
ACB-Photo	7,650	238,163	8	745	No
MAG-Chem	35,409	314,716	14	4,877	Yes
MAG-CS	21,957	193,500	18	7,793	Yes
MAG-Eng	14,927	98,610	16	4,839	Yes
MAG-Med	63,282	1,620,628	17	5,538	Yes

Table 7: Real-world graph summaries.

	GCN		GraphSAGE		GAT		GIN		SkipGCN		*Avg.*	
	NMI	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*
J_{MCP}	59.2	55.6	54.1	46.5	39.7	37.2	49.3	48.9	**60.5**	**58.0**	52.6	49.3
J_{CDMG}	55.9	53.5	45.2	44.9	38.5	37.8	51.6	50.8	52.5	49.9	48.7	47.4
J_{DMoN}	60.9	54.6	54.2	48.3	41.2	37.9	57.9	51.4	56.5	49.3	54.1	48.3
J_{LP}	49.9	44.5	48.2	44.4	40.3	37.1	52.0	48.4	42.4	35.6	46.6	42.0
J_{NOCD}	59.5	55.6	38.5	36.5	36.7	35.0	57.0	53.2	56.0	53.3	49.5	46.7
J_{B}	60.2	61.0	**54.5**	53.8	**42.7**	**41.7**	53.6	**54.8**	58.6	57.8	53.9	**53.8**
J_{P}	59.7	**61.4**	53.9	**54.0**	42.4	41.6	53.8	54.7	58.1	57.1	53.6	53.8
$J_{\text{B-DC}}$	59.8	59.6	53.0	52.9	40.8	39.0	51.8	52.7	57.1	56.2	52.5	52.1
$J_{\text{P-DC}}$	61.2	58.4	53.8	53.4	41.9	40.4	51.3	52.1	59.0	57.7	53.4	52.4
J_{Match}	**61.5**	57.0	51.9	50.5	42.5	40.6	**59.5**	54.5	58.4	53.3	**54.8**	51.2

Table 8: Community detection performance for different GNN architectures. Results are averaged over ten trials for all ten datasets with *non-overlapping* communities. The best scores (NMI and PF1) for each model are in bold.

	GCN		GraphSAGE		GAT		GIN		SkipGCN		*Avg.*	
	NMI	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*	*NMI*	*PF1*
J_{MCP}	38.9	53.8	42.6	52.6	32.9	42.8	36.4	53.4	46.9	54.0	39.6	51.3
J_{CDMG}	13.6	46.8	17.0	54.9	18.5	49.7	18.0	57.4	10.4	40.7	15.5	49.9
J_{DMoN}	37.1	49.9	44.6	53.4	**36.3**	45.4	38.5	54.7	43.3	53.5	40.0	51.4
J_{LP}	14.3	36.0	20.0	49.5	16.9	**50.9**	20.0	48.3	13.9	31.0	17.0	43.1
J_{NOCD}	40.7	48.5	17.2	42.6	16.3	42.8	40.3	57.5	42.3	50.6	31.4	48.4
J_{B}	45.1	58.5	**50.7**	**58.9**	34.9	47.3	37.4	58.8	**51.7**	60.2	**44.0**	56.8
J_{P}	41.6	58.4	50.3	58.3	32.8	45.6	38.7	58.6	48.3	61.4	42.3	56.4
$J_{\text{B-DC}}$	**45.7**	**60.1**	49.5	58.8	34.2	46.8	37.3	58.1	48.4	**63.1**	43.0	**57.4**
$J_{\text{P-DC}}$	43.3	59.3	46.3	56.1	30.1	43.1	34.3	**59.7**	48.4	61.7	40.5	56.0
J_{Match}	43.3	57.2	44.1	54.4	31.3	43.2	**40.7**	59.4	46.9	61.3	41.2	55.1

Table 9: Community detection performance for different GNN architectures. Results are averaged over ten trials for all eight datasets with *overlapping* communities. The best scores (overlapping NMI and PF1) for each model are in bold.