Amortized Probabilistic Detection of Communities in Graphs

Yueqi Wang [*1] **Yoonho Lee** [*2] **Pallab Basu** [3] **Juho Lee** [4] **Yee Whye Teh** [5 6] **Liam Paninski** [7] **Ari Pakman** [8]

Abstract

Learning community structures in graphs has broad applications across scientific domains. While graph neural networks (GNNs) have been successful in encoding graph structures, existing GNN-based methods for community detection are limited by requiring knowledge of the number of communities in advance, in addition to lacking a proper probabilistic formulation to handle uncertainty. We propose a simple framework for amortized community detection, which addresses both of these issues by combining the expressive power of GNNs with recent methods for amortized clustering. Our models consist of a graph representation backbone that extracts structural information and an amortized clustering network that naturally handles variable numbers of clusters. Both components combine into well-defined models of the posterior distribution of graph communities and are jointly optimized given labeled graphs. At inference time, the models yield parallel samples from the posterior of community labels, quantifying uncertainty in a principled way. We evaluate several models from our framework on synthetic and real datasets, and demonstrate improved performance compared to previous methods.

1. Introduction

Community detection (Fortunato, 2010; Yang et al., 2013) is a fundamental problem in network analysis with many applications such as finding communities in social graphs or functional modules in protein interaction graphs. In machine learning, community detection is usually treated either as an unsupervised learning problem, or as posterior inference when a well-defined generative model is assumed.

[*]Equal contribution [1]Google [2]Stanford University [3]University of the Witwatersrand [4]KAIST [5]Oxford University [6]DeepMind [7]Columbia University [8]Ben-Gurion University of the Negev. Correspondence to: Ari Pakman <pakman@bgu.ac.il>.

Accepted by the Structured Probabilistic Inference & Generative Modeling workshop of ICML 2024, Vienna, Austria.

Recent progress in graph neural networks (GNNs) has successfully extended the learning capabilities of deep neural networks to graph-structured data (Bronstein et al., 2017; Hamilton, 2020; Bronstein et al., 2021). Since community structures arise in real-world graphs, the classic problem of community detection can also benefit from the representation learning of GNNs, and this has been indeed the case for unsupervised approaches (Wang et al., 2017; Cavallari et al., 2017; Sun et al., 2019; Jin et al., 2019; Tsitsulin et al., 2020). However, neural models for community detection still face significant challenges, as recently reviewed in (Liu et al., 2020; Jin et al., 2021). A limitation of existing models is the requirement of a fixed or maximum number of communities, typically encoded in the size of a softmax output. This is a long-standing challenge in the field that constrains the model's ability to generalize to new datasets with a varying number of communities. Moreover, the power of GNNs has not been applied to community detection as posterior inference, thus making it impossible to estimate the uncertainty on community assignment or the number of communities in networks, especially in noisy or ambiguous real-world data.

Training a neural network to learn posterior distributions is usually referred to as *amortized inference* (Gershman & Goodman, 2014). Concretely, denote a graph dataset as $\mathbf{x}$, the community labels of its N nodes as $c_{1:N}$, and assume the existence of a joint distribution $p(\mathbf{x}, c_{1:N})$. Amortization consists in training a neural network that takes as input a graph $\mathbf{x}$ and outputs a distribution over community assignments $p(c_{1:N}|\mathbf{x})$. In exchange for the initial cost of training a neural model, amortized inference offers several benefits compared to either unsupervised approaches or other posterior inference methods.

Compared to unsupervised neural models for community discovery, amortized approaches are more time-efficient at test time, as they only require a forward pass evaluation on a pre-trained model, whereas unsupervised models typically go through an iterative optimization process for every test example (Wang et al., 2017; Cavallari et al., 2017; Sun et al., 2019; Jin et al., 2019; Tsitsulin et al., 2020). Moreover, the generic inductive biases encoded in unsupervised models might not be optimal for datasets with different structures, and amortized approaches can help by incorporating prior knowledge through model training.

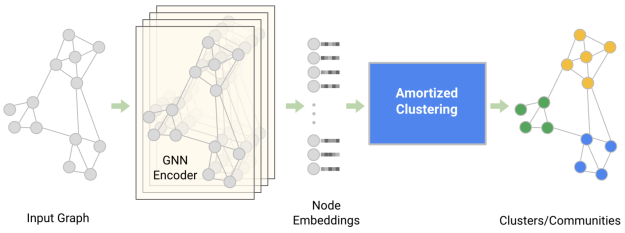

Figure 1: **Amortized Community Detection.**

Compared to other posterior inference methods, the benefits of amortization are threefold. First, well-trained models can produce i.i.d. posterior samples that match the accuracy of time-consuming Markov chain Monte Carlo (MCMC) methods at a fraction of the time (Pakman et al., 2020). Second, MCMC methods, as well as the faster but less accurate variational methods (Blei et al., 2017), usually require explicit forms for $p(c_{1:N})$ and $p(\mathbf{x}|c_{1:N})$ in order to sample or approximate the posterior $p(c_{1:N}|\mathbf{x})$, while amortization only requires *samples* from the joint generative model $p(\mathbf{x}, c_{1:N})$, but not the putative model itself. Thus real-world labeled datasets can be easily incorporated into a probabilistic inference setting without the need to fit a generative model. Finally, neural models that represent $p(c|\mathbf{x})$ yield explicit values for the probability of each sample c, a quantity usually unavailable in MCMC. Thus when a single community labeling for a graph is required, one can simply pick the highest probability sample.

In this work we present an amortized framework for community discovery that combines GNNs with improved versions of amortized clustering architectures that naturally accommodate varying cluster numbers (Lee et al., 2019b; Pakman et al., 2020), as illustrated in Figure 1. Our models are trained with labeled datasets containing varying numbers of communities, and yield samples from the posterior over community labels for test graphs of any size.

2. Related Works

Neural models for community detection. As mentioned above, almost all previous works on GNN-based community detection adopt an unsupervised learning approach. The only previous supervised learning work addressing this task is (Chen et al., 2018), which proposed two encodings for graph nodes specialized for community detection tasks (referred below as GNN and LGNN). For a fixed number K of clusters, the models output for each node i a softmax $\phi_{i,c}$ for $c = 1 \dots K$. The objective function is

$$I(\theta) = \min_{\pi \in \mathcal{S}_K} \sum_{i \in \mathcal{D}} \log \phi_{i,\pi(c_i)} \,, \tag{1}$$

where $\mathcal{D}$ indexes the dataset and $\mathcal{S}_K$ is the permutation group over the K labels. Thus apart from fixing a maximum K in advance, the models incur the cost of evaluating $K!$ terms, which makes them impractical for $K > 8$ (see Figure 7 below). More importantly, treating community discovery as node classification ignores an important inductive bias of the problem, as explained below in Section 4.

A related supervised learning problem was studied in (Dwivedi et al., 2020) as a benchmark for GNN architectures. But the task here is to find the members of each community *given a known initial node* for each community, and is thus not directly comparable with our more difficult generic setting.

Amortized Clustering. The amortized clustering models we adopt and extend in this work, reviewed in the next section, differ from previous works on supervised clustering (Finley & Joachims, 2005; Al-Harbi & Rayward-Smith, 2006), attention-based clustering (Coward et al., 2020; Ienco & Interdonato, 2020) and neural network-based clustering (reviewed in (Du, 2010; Aljalbout et al., 2018; Min et al., 2018)), as these works focus on learning data features or similarity metrics as inputs to traditional clustering algorithms, while we instead model the posterior distribution of a generative model of clusters.

Supervised amortized probabilistic clustering was studied in (Le et al., 2017; Lee et al., 2019a; Kalra et al., 2020) for Gaussian mixtures with a fixed or bounded number of components. In those papers, the outputs of the network are the mixture parameters. This differs from the models we use in this work, which instead output the cluster labels of each data point, and are not restricted to mixtures of Gaussians.

3. Background

3.1. Generative Models of Clusters

Our approach to community discovery is guided by its connection to generative models of clustering (McLachlan & Basford, 1988). Let us denote the cluster labels as random variables c_i, and assume a generative process

$$\begin{aligned} N &\sim p(N) \\ \alpha_i &\sim p(\alpha_i), \quad i = 1, 2 \\ c_1 \dots c_N &\sim p(c_1, \dots, c_N | \alpha_1) \,. \end{aligned} \tag{2}$$

The distribution in (2) is an exchangeable clustering prior and α_1, α_2 are hyperparameters. We define the integer random variable K as the number of distinct cluster indices c_i. Note that K can take any value $K \leq N$, thus allowing for Bayesian nonparametric priors such as the Chinese Restaurant Process (CRP) (see (Rodriguez & Mueller, 2013) for a review). This framework also encompasses cluster numbers $K < B$ for fixed B, such as Mixtures of Finite Mixtures (Miller & Harrison, 2018).

Given the cluster labels, standard mixture models assume

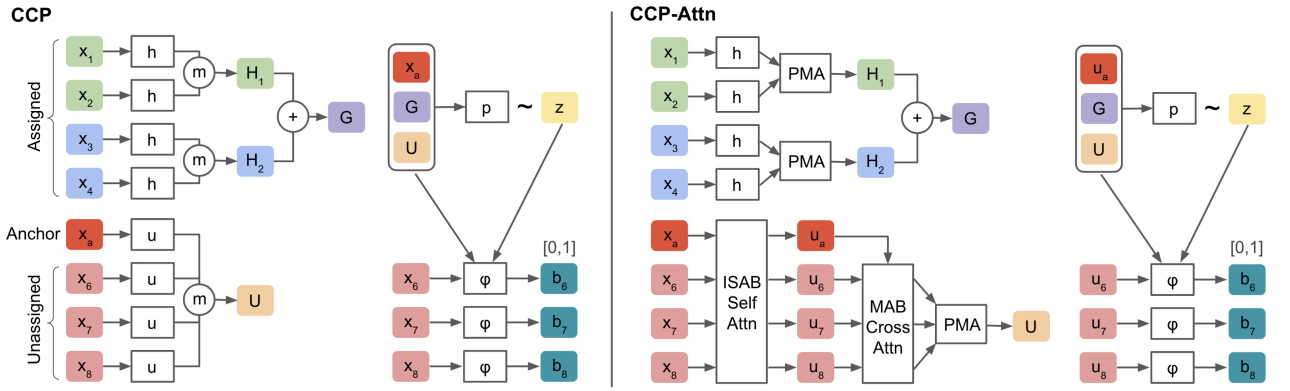

Figure 2: **CCP and CCP-Attn.** *Left:* Architecture of the CCP model (Pakman et al., 2020) for clusterwise amortized clustering. *Right:* Our proposed modification, CCP-Attn., where the mean aggregations ⓜ used by CCP (see equation (7)) are replaced by Set Transformer attention modules from (Lee et al., 2019a). See Appendix B.4 for details.

N observations $\mathbf{x} = \{x_i\}$ generated as

$$\begin{aligned} \mu_1 \dots \mu_K &\sim p(\mu_1, \dots, \mu_K | \alpha_2) \\ x_i &\sim p(x_i | \mu_{c_i}) \quad \text{for each} \quad i = 1 \dots N. \end{aligned} \tag{3}$$

Here μ_k controls the distribution of the k-th cluster, e.g. as the mean and covariance of a Gaussian mixture component.

3.2. Review of Amortized Clustering Models

Given N observations $\mathbf{x} = \{x_i\}$, amortized clustering methods parameterize and learn a mapping from $\mathbf{x}$ to a distribution over the indices $\{c_i\}$. In this work we consider and compare three recent amortization models:

1. Pointwise expansion (NCP Model). Given N data points $\mathbf{x} = \{x_i\}$, we can sequentially expand the posterior distribution of labels as

$$p(c_{1:N}|\mathbf{x}) = p(c_1|\mathbf{x})p(c_2|c_1, \mathbf{x}) \dots p(c_N|c_{1:N-1}\mathbf{x}). \tag{4}$$

A neural architecture to model these factors, called the Neural Clustering Process (NCP), was proposed in (Pakman et al., 2020), and requires $O(N)$ forward evaluations for a full sample of cluster labels. Note that the range of values that c_n can take is not fixed, since it depends on the previously assigned labels $c_{1:n-1}$. Therefore, the multinomial distribution $p(c_n|c_{1:n-1}\mathbf{x})$ is not represented using a standard softmax output, but a novel 'variable-input softmax' introduced in (Pakman et al., 2020).

2. Clusterwise expansion (CCP Model). An equivalent representation of the cluster labels $c_{1:N}$ is given by the collection of K sets $\mathbf{s}_{1:K}$, where each $\mathbf{s}_k$ contains the indices of points belonging to cluster k. For example, labels $c_{1:6} = (1, 1, 2, 1, 2, 1)$ are equivalent to $\mathbf{s}_1 = (1, 2, 4, 6)$, $\mathbf{s}_2 = (3, 5)$. Thus we can expand the posterior as

$$p(\mathbf{s}_{1:K}|\mathbf{x}) = p(\mathbf{s}_1|\mathbf{x})p(\mathbf{s}_2|\mathbf{s}_1, \mathbf{x}) \dots p(\mathbf{s}_K|\mathbf{s}_{1:K-1}, \mathbf{x}). \tag{5}$$

with $p(c_{1:N}|\mathbf{x}) = p(\mathbf{s}_{1:K}|\mathbf{x})$. A neural model for these factors is the Clusterwise Clustering Process (CCP) (Pakman et al., 2020) (Figure 2). To sample from $p(\mathbf{s}_k|\mathbf{s}_{1:k-1}, \mathbf{x})$ we iterate two steps: (i) sample uniformly the first element x_a of $\mathbf{s}_k$ (called anchor) from the available points, (ii) choose which points join x_a to form $\mathbf{s}_k$ by sampling from

$$p(\mathbf{b}_k|x_a, \mathbf{s}_{1:k-1}, \mathbf{x}), \tag{6}$$

where $\mathbf{b}_k \in \{0, 1\}^{m_k}$ is a binary vector associated with the m_k remaining data points $\{x_{q_i}\}_{i=1}^{m_k}$. This distribution depends both on these remaining points and on the assigned clusters $\mathbf{s}_{1:k-1}$ in permutation-symmetric ways, which can be respectively captured by symmetric encodings of the form (Zaheer et al., 2017)

$$U = \frac{1}{m_k} \sum_{i=1}^{m_k} u(x_{q_i}), \quad G = \sum_{j=1}^{k-1} g\left(\frac{1}{|\mathbf{s}_j|} \sum_{i \in \mathbf{s}_j} h(x_i) \right), \tag{7}$$

where u, h, g are neural networks with vector outputs. Moreover, the binary distribution (6) satisfies a form of conditional exchangeability (Pakman et al., 2020) and can be represented as

$$\begin{aligned} &p(\mathbf{b}_k|x_a, \mathbf{s}_{1:k-1}, \mathbf{x}) \simeq \\ &\int d\mathbf{z}_k \prod_{i=1}^{m_k} \varphi(b_i|\mathbf{z}_k, U, G, x_a, x_{q_i}) \mathcal{N}(\mathbf{z}_k|U, G, x_a). \end{aligned} \tag{8}$$

Here $\varphi(b_i = 1|\cdot)$ is a neural network with a sigmoid output, and the latent variable $\mathbf{z}_k \in \mathbb{R}^{d_z}$ is a Gaussian, with mean and variance parametrized by neural networks with input (U, G, x_a). Since b_i's in (8) are independent given $\mathbf{z}_k$, after sampling $\mathbf{z}_k$ all the b_i's can be sampled in parallel. Thus while a full sample of $\mathbf{s}_{1:K}$ has cost $O(N)$, the heaviest computational burden, from network evaluations, scales as $O(K)$, as each factor $p(\mathbf{s}_k|\mathbf{s}_{1:k-1}, \mathbf{x})$ in (5) needs $O(1)$ forward calls. Probability estimates for each sampled clustering configuration are provided in (Pakman et al., 2020). The CCP architecture is illustrated in Figure 2, left.

3. Non-probabilistic clusterwise expansion (DAC Model). The Deep Amortized Clustering (DAC) (Lee et al., 2019b) model is based on the expansion (5), but does not fully preserve the inductive biases of the generative model. In its simplest version, it learns a binary classifier similar to (6), but assumes a form

$$p\left(\mathbf{b}_k \mid \mathbf{x}, x_a\right) \simeq \prod_{i=1}^{m_k} p\left(b_i \mid x_{i:m_k}, x_a\right) , \tag{9}$$

i.e., it ignores previously sampled clusters $\mathbf{s}_{1:k-1}$ and the dependencies captured by $\mathbf{z}_k$ in (8).

All three models are trained with labelled samples $(c_{1:N}, \mathbf{x})$ by optimizing the model likelihood (for NCP and DAC) or an evidence lower bound (ELBO) thereof (for CCP). See (Pakman et al., 2020; Lee et al., 2019b) for details.

3.3. Review of Graph Neural Networks

The class of GNNs that we will consider are the widely used graph convolutional networks (GCNs), which produce a feature vector $h_i^L \in \mathbb{R}^{d_h}$ associated to each node i in a graph according to the formula

$$h_i^{\ell+1} = \sigma\left(W_1^\ell h_i^\ell + \sum_{j\in\mathcal{N}_i} \eta_{ij}^\ell W_2^\ell h_j^\ell\right), \tag{10}$$
$$h_i^\ell \in \mathbb{R}^{d_h}, \quad W_{1,2}^\ell \in \mathbb{R}^{d_h\times d_h}, \quad \ell = 0\dots L-1,$$

where $\mathcal{N}_i$ is the set of neighbours of node i, and $\eta_{ij}^\ell = f^\ell(h_i^\ell, h_j^\ell)$ is a scalar or vector function (multiplied elementwise) that models the anisotropy of the encoding. Note that the update equation (10) is independent of the graph size and is a form of convolution since it shares the same weights across the graph. Each node requires also an initial input vector h_i^0, which can contain, e.g., node attributes. For recent book-length overviews of this vast topic see (Hamilton, 2020; Bronstein et al., 2021).

4. Amortized Community Detection

4.1. Generative Model of Graphs with Communities

For communities in graphs, we assume community labels for each node are generated as in the clustering prior (2), followed by a generative model of edge data $\mathbf{y} = \{y_{ij}\}_{i,j=1}^N$ (e.g. direction or strength), and possibly node features $\mathbf{f} = \{f_i\}_{i=1}^N$. For example, a popular generative model (without node features) is

$$\phi_{k_1,k_2} \sim p(\phi|\beta) \quad k_1 \le k_2 \tag{11}$$
$$y_{ij} \sim \text{Bernoulli}(\phi_{c_i,c_j}), \quad i \le j, \quad i,j = 1\dots N \tag{12}$$

where $k_1, k_2 = 1\dots K$ and the y_{ij}'s are binary variables representing the presence or absence of an edge in the graph. Both the stochastic block model (SBM) (Holland et al., 1983) and the single-type Infinite Relational Model (Kemp et al., 2006; Xu et al., 2006) use variants of the generative model (11)-(12).

4.2. Combining GNN with Amortized Clustering

Our proposal in this work, illustrated in Figure 1, is to use a GNN to map observations $\mathbf{y}, \mathbf{f}$ to an embedding vector for each node,

$$x_i = h_i^L(\mathbf{y}, f_i), \qquad i = 1\dots N. \tag{13}$$

We assume these node embeddings are approximate sufficient statistics for the posterior over labels,

$$p(c_{1:N}|\mathbf{y},\mathbf{f}) \simeq p(c_{1:N}|x_{1:N}), \tag{14}$$

and use node embeddings $x_{1:N}$ as inputs to the amortized clustering modules reviewed in Section 3.2.

For the initial node features $f_{1:N}$ as inputs to the GNN encoder, we choose the method of Laplacian eigenvector positional encoding (Belkin & Niyogi, 2003; Dwivedi et al., 2020), which takes the m smallest non-trivial eigenvectors of the graph Laplacian matrix and encodes the graph positional information for each node.

Given labeled datasets of the form $(c_{1:N}, \mathbf{y}, \mathbf{f})$, both neural network modules (GNN and amortized clustering) are trained end-to-end by plugging in the objective function of the amortized model and the GNN-dependent inputs x_i. Thus our proposed framework encourages the GNN to output node features that compactly represent the community structure, and the amortized clustering module uses such compact node features to identify communities.

We visualize the progression of label assignments in a graph with community structure for the two different posterior expansions of the amortized clustering module. Figure 3 and Figure 4 respectively illustrate the node-wise expansion used by NCP (eqn. 4) and the community-wise expansion used by CCP and DAC (eqn. 5). We note that the simplifying assumptions made by DAC about label independence are similar to those made by models that treat community discovery as classification (such as LGNN (Chen et al., 2018) or CLUSTER (Dwivedi et al., 2020)).

4.3. Adding Attention to Amortized Clustering

To increase the expressivity of the amortized clustering models, we added attention modules to the NCP and CCP models by replacing mean and sum aggregation steps with variants of multi-head attention (MHA) blocks (Vaswani et al., 2017). For DAC (Lee et al., 2019b), the original formulation already has attention modules. More concretely, we used three MHA-based modules with learnable parameters defined in (Lee et al., 2019a) that act on sets, named Multihead

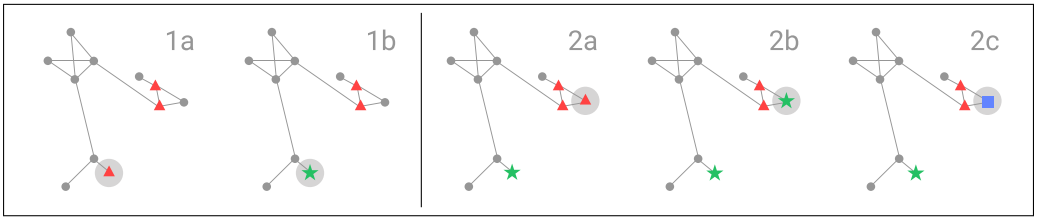

Figure 3: **Node-wise sampling (NCP Model).** *1a-1b*: Two nodes have been already assigned to the same community ($c_1 = c_2 = 1$, red triangles) and a randomly selected new node is sampled from $p(c_3|c_{1:2}, \mathbf{x})$ to choose whether it joins them (1a, $c_3 = 1$), or creates a new community (1b, $c_3 = 2$, green star). *2a-2c*: The previous node started its own community, and the next random node is sampled from $p(c_4|c_{1:3}, \mathbf{x})$ to choose whether it joins any of the existing communities (2a, $c_4 = 1$; 2b, $c_4 = 2$) or creates a new community (2c, $c_4 = 3$, blue square). The procedure is repeated until all nodes have are sampled.

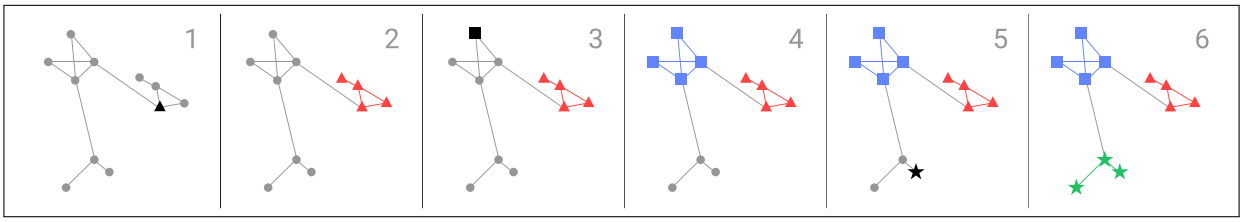

Figure 4: **Community-wise sampling (CCP and DAC Models).** (1) The first element of community $\mathbf{s}_1$ (black triangle) is sampled uniformly, and the available points (grey dots) are queried to join. (2) The first community $\mathbf{s}_1$ is formed (red triangles). (3) The first element of $\mathbf{s}_2$ (black square) is sampled uniformly from unassigned points. (4) The second community $\mathbf{s}_2$ is formed (blue squares). (5)-(6) We repeat this procedure until no unassigned points are left. In CCP, the binary queries are correlated, but become independent conditioned on a latent vector, thus allowing parallel sampling (eq.(8)).

Attention Block (MAB), Pooling by Multihead Attention (PMA), and Induced Self-Attention Block (ISAB).

We call the amortized models with attention NCP-Attn and CCP-Attn. Figure 2, right, illustrates how CCP-Attn modifies CCP by replacing the mean operations in equation (7) by attention modules. For NCP, attention-based aggregation is harder to incorporate due to the $O(N)$ forward evaluations. Instead, we replaced the h and u functions in (7) with ISAB self-attention layers across all input points, so that it is evaluated only once before the $O(N)$ iterations. We describe the details of the attention modules and their use in the NCP and CCP architectures in Appendix B.2.

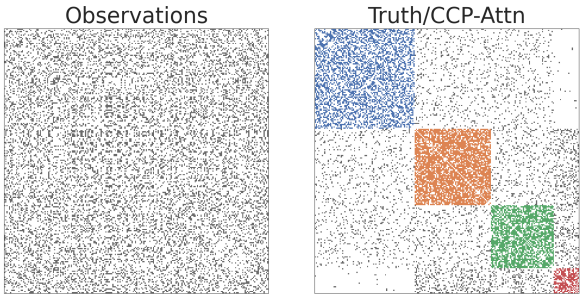

Figure 5: **General SBM.** *Left:* Observations ($N = 222$). *Right:* Exact community recovery by CCP-Attn.

Table 1: **Comparison community detection methods.** The column 'Pr.' indicates whether the method is probabilistic. ($*$): LGNN takes only one evaluation because it assumes fixed or maximum K and is not directly comparable.

Method	Learning	K Constraints	Pr.	Cost
DMoN (Tsitsulin et al., 2020)	Unsupervised	Max K	✗	N/A
LGNN/GNN (Chen et al., 2018)	Supervised	$K < 8$	✗	1^*
GCN-DAC (Lee et al., 2019b) (ours)	Supervised	Varying K	✗	$O(K)$
GCN-NCP (Pakman et al., 2020) (ours)	Supervised	Varying K	✓	$O(N)$
GCN-CCP (Pakman et al., 2020) (ours)	Supervised	Varying K	✓	$O(K)$

5. Experiments

We present several experiments to illustrate our framework. Comparisons are made with LGNN/GNN (Chen et al., 2018), a previous supervised model for community detection, and DMoN (Tsitsulin et al., 2020), a recent unsupervised neural model for community detection with state-of-the-art performance compared to other unsupervised approaches. Table 1 compares all the models we considered.

In the examples below, the reported single estimates from our NCP and CCP-based models correspond to the maximum a posteriori (MAP) sample maximizing the probability

Table 2: **Clustering SBM using CCP-Attn with different input embeddings and GCN encoders.** The SBM data contains $1 \sim 16$ communities. Pos Enc: positional encoding; Rand Feat: random features.

Input	GCN	AMI$_{\times 100}$	ARI$_{\times 100}$	ELBO
Rand Feat.	GraphSAGE	57.0$_{\pm 0.5}$	54.4$_{\pm 0.9}$	-120$_{\pm 3}$
Rand Feat.	GatedGCN	66.5$_{\pm 1.9}$	65.5$_{\pm 2.5}$	-93$_{\pm 2}$
Pos Enc.	GraphSAGE	82.8$_{\pm 0.2}$	82.1$_{\pm 0.1}$	-45$_{\pm 0.1}$
Pos Enc.	GatedGCN	**89.6**$_{\pm 0.4}$	**88.5**$_{\pm 0.6}$	**-23**$_{\pm 0.4}$

$p(c_{1:N}|\mathbf{x})$, estimated from multiple GPU-parallelized posterior samples. A default sampling size of 15 is used unless noted otherwise. We measure similarity between inferred and true community labels using the Adjusted Mutual Information (AMI) (Vinh et al., 2010) and/or Adjusted Rand Index (ARI) (Hubert & Arabie, 1985) scores, which take values in $[0, 1]$, with 1 corresponding to perfect matching. Other metrics are indicated in each case.

To study a generic form of community detection, in all our examples the data is given by undirected edge connectivity, and nodes had no attributes. Details of the neural architectures and experiment setup appear in the Appendix. Code to reproduce our experiments is available at `https://github.com/aripakman/amortized_community_detection`.

5.1. Datasets

We consider two main types of datasets for evaluating our community detection models.

General SBM. The SBM generative model is eqs.(11)-(12) and generated with $N \sim \text{Unif}[50, 350]$, $c_1 \dots c_N \sim \text{CRP}(\alpha)$, $p = \phi_{k_1,k_2} \sim \text{Beta}(6, 4)$ for $k_1 = k_2$, and $q = \phi_{k_1,k_2} \sim \text{Beta}(1, 7)$ for $k_1 \neq k_2$. Here CRP(α) is a Chinese Restaurant Process with concentration $\alpha = 3.0$. The graphs contain varying ($K = 1 \sim 16$) numbers of communities with varied connection probabilities, as illustrated in Figure 5. The SBM data tests the models' ability to learn from a generative model.

Real-world SNAP datasets. The SNAP datasets (Leskovec & Krevl, 2014) contains real-world networks from social connections, collaborations, and consumer products. Unlike the SBM, we lack here an explicit generative model, but that is not a problem since all we need for training are ground-truth community labels. We used two SNAP datasets (DBLP, Youtube) and followed the data preparation of (Yang & Leskovec, 2015) and (Chen et al., 2018) to extract subgraphs composed of 2 to 4 non-overlapping communities.

5.2. GNN Encoders and Input Node Features

To find an effective GNN encoder, we considered two GCN variants: (i) the isotropic GraphSAGE (Hamilton et al., 2017) (with $\eta_{ij}^{\ell} = 1$ in (10)) and (ii) the anisotropic GatedGCN (Bresson & Laurent, 2018), shown to perform well on node classification benchmarks (Dwivedi et al., 2020). As input features of the nodes we used a 20-dimensional Laplacian eigenvector positional encoding (Belkin & Niyogi, 2003; Dwivedi et al., 2020), and compared it with a baseline of 20-dimensional random vector sampled from $\mathcal{N}(0, 1)$. Using the CCP-Attn amortized clustering model and training on the SBM dataset, we show in Table 2 that GatedGCN and positional encoding consistently perform better than GraphSAGE and random input features, and we use them for other experiments below.

5.3. Performance on SBM and SNAP datasets

Table 3 summarizes the results of amortized community detection on the general SBM and SNAP datasets. Since DMoN and LGNN require hard-coding a fixed K in the neural architectures, we experimented with different K values higher or equal to the maximum number of communities.

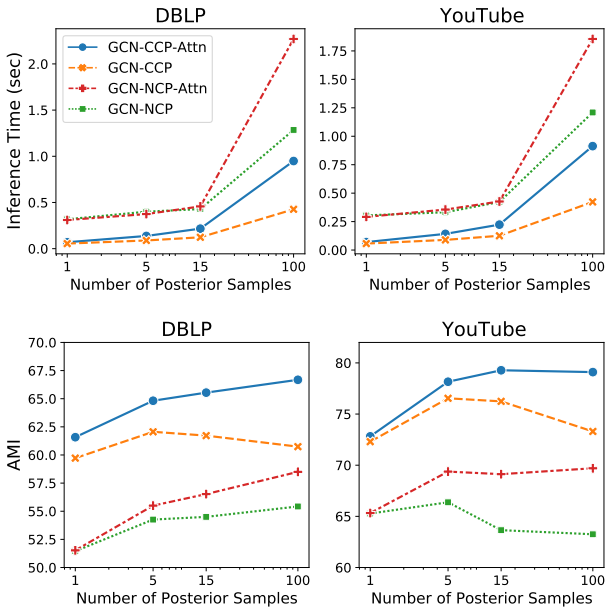

Figure 6: **Inference time and AMI scores as a function of the number of posterior samples.** Results for two SNAP datasets with $K = 2 \sim 4$. *Top:* CCP is faster than NCP not only for changing N or K, but also as a function of the number of posterior samples. *Bottom:* AMI scores. Note that when the model includes attention, the clustering quality improves with more samples; this trend is weaker or absent without attention. Each point is the mean of 4 runs.

Table 3: **Clustering performance on synthetic SBM and real-world SNAP datasets.** Each dataset contains either fixed or varying numbers of communities as denoted by K. For models that require fixing a maximum K, results for different K values are shown. The AMI and ARI scores are multiplied by 100. Times are in the unit of seconds. Means and standard deviations of metrics are from 3 – 6 independently trained models (DMoN was run once on each dataset). The standard deviation of inference time is small and thus omitted. OOM: out-of memory.

		SBM ($K = 1 \sim 16$)			DBLP ($K = 3$)		DBLP ($K = 2 \sim 4$)			Youtube ($K = 3$)		Youtube ($K = 2 \sim 4$)		
Model	Max K	AMI	ARI	Time	AMI	ARI	AMI	ARI	Time	AMI	ARI	AMI	ARI	Time
DMoN (Tsitsulin et al., 2020)	16	60.0	46.7	12.1	52.8	33.4	39.7	25.3	10.7	40.6	18.2	32.4	14.8	11.4
	4	-	-	-	54.6	47.1	45.1	36.5	10.0	50.2	43.0	40.2	29.3	10.1
LGNN (Chen et al., 2018)	7	OOM	OOM	OOM	-	-	$54.2_{\pm 2.1}$	$51.2_{\pm 2.3}$	0.44	-	-	$68.7_{\pm 0.7}$	$71.3_{\pm 1.1}$	0.28
	4	-	-	-	-	-	$52.7_{\pm 2.2}$	$49.4_{\pm 2.3}$	0.42	-	-	$68.4_{\pm 1.4}$	$71.3_{\pm 1.1}$	0.27
	3	-	-	-	$66.8_{\pm 1.3}$	$62.7_{\pm 1.6}$	-	-	-	$76.7_{\pm 0.8}$	$78.4_{\pm 0.8}$	-	-	-
GNN (Chen et al., 2018)	7	$76.2_{\pm 0.9}$	$75.9_{\pm 1.2}$	0.15	-	-	$51.5_{\pm 2.5}$	$48.4_{\pm 2.5}$	0.035	-	-	$67.3_{\pm 3.8}$	$69.5_{\pm 4.0}$	0.033
	4	-	-	-	-	-	$55.4_{\pm 3.3}$	$52.2_{\pm 4.1}$	0.036	-	-	$67.5_{\pm 1.2}$	$69.7_{\pm 1.4}$	0.033
	3	-	-	-	$66.3_{\pm 1.7}$	$62.7_{\pm 1.7}$	-	-	-	$76.2_{\pm 2.6}$	$78.0_{\pm 2.0}$	-	-	-
GCN-NCP	-	$89.6_{\pm 0.2}$	$88.3_{\pm 0.3}$	0.88	$74.3_{\pm 2.1}$	$71.0_{\pm 2.1}$	$54.8_{\pm 0.3}$	$51.0_{\pm 0.3}$	0.54	$84.5_{\pm 0.9}$	$84.6_{\pm 0.9}$	$64.7_{\pm 3.4}$	$64.2_{\pm 3.3}$	0.43
GCN-NCP-Attn	-	**89.9**$_{\pm 0.4}$	**88.9**$_{\pm 0.6}$	1.0	$82.1_{\pm 0.6}$	$78.7_{\pm 0.5}$	$56.4_{\pm 2.1}$	$52.4_{\pm 2.0}$	0.47	$84.3_{\pm 0.9}$	$84.5_{\pm 1.0}$	$69.2_{\pm 4.1}$	$68.9_{\pm 4.3}$	0.43
GCN-DAC	-	$87.1_{\pm 0.1}$	$87.1_{\pm 0.1}$	0.16	$79.2_{\pm 1.4}$	$74.4_{\pm 1.8}$	$60.3_{\pm 0.7}$	$55.9_{\pm 0.8}$	0.073	$82.6_{\pm 0.5}$	$82.6_{\pm 0.6}$	$73.7_{\pm 1.3}$	$73.5_{\pm 1.5}$	0.081
GCN-CCP	-	$88.8_{\pm 0.5}$	$87.3_{\pm 0.7}$	0.3	$82.1_{\pm 0.5}$	$78.6_{\pm 0.6}$	$62.4_{\pm 3.7}$	$57.6_{\pm 3.4}$	0.12	$83.7_{\pm 0.8}$	$83.8_{\pm 0.7}$	$76.2_{\pm 1.9}$	$76.2_{\pm 2.1}$	0.13
GCN-CCP-Attn	-	$89.6_{\pm 0.4}$	$88.5_{\pm 0.6}$	0.59	**89.2**$_{\pm 0.4}$	**79.7**$_{\pm 0.5}$	**65.3**$_{\pm 1.2}$	**59.7**$_{\pm 1.8}$	0.22	**85.5**$_{\pm 0.3}$	**85.7**$_{\pm 0.4}$	**79.2**$_{\pm 0.4}$	**79.3**$_{\pm 0.4}$	0.22

The result shows that combining GCN with amortized clustering (e.g. GCN-CCP) learns more accurate community discovery than LGNN/GNN (Chen et al., 2018) in terms of AMI and ARI, and that adding attention modules to CCP gives rise to the best-performing model GCN-CCP-Attn.

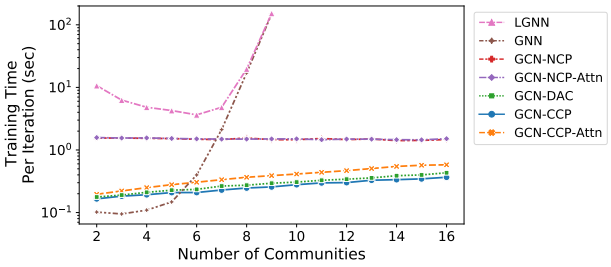

Figure 7: **Training times of amortized models.** Per-iteration training time as a function of the number of communities. The models are trained on the same GPU with batch size of 1 on a SBM dataset with 160 nodes and varying numbers of equal-size communities ($p = 0.3$, $q = 0.1$).

In the SBM dataset with $1 \sim 16$ communities, both GCN-NCP and GCN-CCP models achieved similarly good performance, and GCN-NCP outperforms by an small margin. Due to the $O(K!)$ loss function, LGNN/GNN can only train on graphs with up to 7 communities, thus cannot learn community detection at $K \geq 8$.

The real-world SNAP datasets are composed of subgraphs with either fixed ($K = 3$) or varying ($K = 2 \sim 4$) numbers of communities. On all SNAP benchmarks, GCN-CCP provides consistently higher AMI and ARI than other models, and GCN-CCP-Attn further improves performance by a significantly margin. We attribute this performance gain on SNAP datasets to the expressive power of attention.

5.4. Time Measures and MAP Estimates

Table 3 illustrates the benefits of amortization at test time, as our models run more than 10x faster than the unsupervised model (DMoN). The training time (5~10 hrs) is worth if the number of test examples is on the order of thousands.

Figure 6, top, shows that CCP models scale better than NCP as a function of not only the number of data points, but also the number of posterior samples. Figure 6, bottom, shows that for models with attention, the clustering quality improves with more samples, indicating a better fit of the learned probabilistic model to the data distribution.

Figure 7 reports per-iteration training time. The steep time increase w.r.t. K limits LGNN/GNN to graphs with less than 8 communities. The moderate growth w.r.t K of DAC/CCP/NCP illlustrates the benefits of our architectures.

5.5. Recovery Thresholds in Log-Degree SBM

We consider next symmetric SBMs with K communities of equal size. The connection probability is $p = a\log(N)/N$ within and $q = b\log(N)/N$ between communities. The expected degree of a node is known to be $O(\log N)$. This is an interesting regime, since for large N it was shown using information-theoretic arguments that the maximum likelihood estimate of the c_i's recovers exactly the community structure with high probability for $|\sqrt{a} - \sqrt{b}| > \sqrt{K}$ and fails for $|\sqrt{a} - \sqrt{b}| < \sqrt{K}$ (Abbe et al., 2015; Mossel et al., 2014; Abbe & Sandon, 2015a;b).

We trained GCN-CCP and GCN-CCP+Attn networks with samples generated with $K \in \{2, 3, 4\}$, $N \in [300, 600]$ and

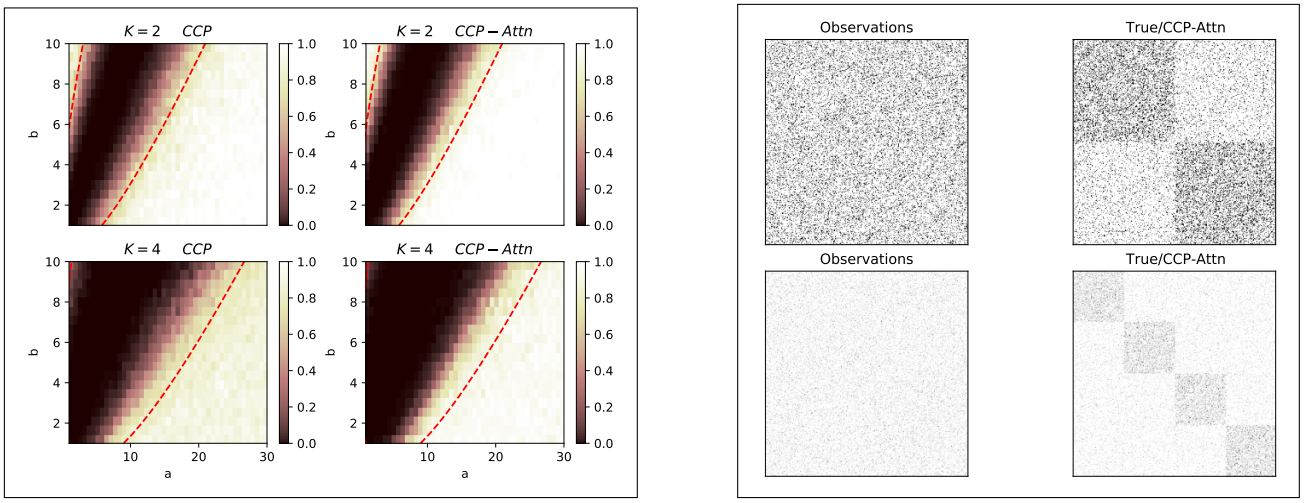

Figure 8: **Log-degree symmetric SBM.** Each network in this family has K communities of equal size N/K, as illustrated in the right. The connection probability is $p = a\log(N)/N$ within each community and $q = b\log(N)/N$ between communities. *Left:* Dashed red curves indicate the threshold $|\sqrt{a} - \sqrt{b}| = \sqrt{K}$, which separates regions of possible/impossible recovery with high probability using maximum-likelihood at large N (Abbe et al., 2015; Mossel et al., 2014; Abbe & Sandon, 2015a;b). *Left above:* Mean AMIs over 40 test datasets with $N = 300, K = 2$. The mean AMI averaged over the recoverable regions is 0.970 for GCN-CCP and 0.997 for GCN-CCP+Attn. *Left below:* Similar for mean of 10 datasets with $N = 600, K = 4$. The mean AMI averaged over the recoverable regions is 0.858 for GCN-CCP and 0.942 for GCN-CCP+Attn. Note that the advantage of GCN-CCP-Attn over GCN-CCP is more prominent for higher K, and that the thresholds are crossed in our finite N case. *Right:* Examples of observed adjacency matrices and successful exact recovery by GCN-CCP-Attn, for $N = 300$, $K = 2$, $a = 15$, $b = 5$ (*above*), and $N = 600$, $K = 4$, $a = 15$, $b = 4$ (*below*).

(a, b) sampled uniformly from $[1, 30] \times [1, 10]$. Figure 8 shows mean test AMI scores and examples of exact recovery. Note in particular that improvement due to the presence of the attention modules is more prominent as K increases.

5.6. Uncertainty quantification

An advantage of our fully probabilistic model is the ability to quantify the uncertainty of inferred quantities. We illustrate this in Figure 10 (Appendix), which shows the mean and and standard deviation of the number of clusters K in a family of SBM models (see details in the Figure's caption).

5.7. Calibration

In order to probe how well calibrated the learned posterior distributions are, we calculated the expected calibration error (ECE) metric (Guo et al., 2017) for classification applied to the prediction of the number of clusters K. ECE is defined as $\mathrm{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\mathrm{acc}(B_m) - \mathrm{conf}(B_m)|$, where n is the number of samples and each B_m is one of M equally spaced bins, where B_m is the set of indices whose confidence is inside the interval $I_m = \left(\frac{m-1}{M}, \frac{m}{M}\right]$. In Table 4, we compare the ECE for two of our models. We sampled 15 community assignments from the posterior of each trained model and used the distribution of predictions to measure confidence. As expected, the CCP model performs better than DAC, since its architecture encodes the correct probabilistic inductive biases.

Table 4: **Expected Calibration Error on SNAP datasets.** Means and st. dev. of ECE from 5 independent runs.

	DBLP	Youtube
GCN-DAC	$0.7406_{\pm 0.1325}$	$0.7932_{\pm 0.0716}$
GCN-CCP	$\mathbf{0.1579}_{\pm \mathbf{0.0295}}$	$\mathbf{0.1804}_{\pm \mathbf{0.0286}}$

6. Conclusion

We have introduced a novel framework for efficiently detecting community structures in graph data, building on recent advances in graph neural networks and amortized clustering. Our experiments have shown that our proposed method outperforms previous methods for community detection on many benchmarks. Possible future directions include incorporating new combinations of attention modules, learning SBMs in the weak recovery regime (Abbe, 2018), and dealing with overlapping communities.

References

Abbe, E. Community Detection and Stochastic Block Models. *Foundations and Trends® in Communications and Information Theory*, 14(1-2):1–162, 2018.

Abbe, E. and Sandon, C. Community detection in general stochastic block models: Fundamental limits and efficient algorithms for recovery. In *2015 IEEE 56th Annual Symposium on Foundations of Computer Science*, pp. 670–688. IEEE, 2015a.

Abbe, E. and Sandon, C. Recovering communities in the general stochastic block model without knowing the parameters. In *Advances in neural information processing systems*, pp. 676–684, 2015b.

Abbe, E., Bandeira, A. S., and Hall, G. Exact recovery in the stochastic block model. *IEEE Transactions on Information Theory*, 62(1):471–487, 2015.

Al-Harbi, S. H. and Rayward-Smith, V. J. Adapting k-means for supervised clustering. *Applied Intelligence*, 24 (3):219–226, 2006.

Aljalbout, E., Golkov, V., Siddiqui, Y., and Cremers, D. Clustering with Deep Learning: Taxonomy and New Methods. *arXiv preprint arXiv:1801.07648*, 2018.

Belkin, M. and Niyogi, P. Laplacian eigenmaps for dimensionality reduction and data representation. *Neural computation*, 15(6):1373–1396, 2003.

Blei, D. M., Kucukelbir, A., and McAuliffe, J. D. Variational inference: A review for statisticians. *Journal of the American statistical Association*, 112(518):859–877, 2017.

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2018.

Bronstein, M. M., Bruna, J., LeCun, Y., Szlam, A., and Vandergheynst, P. Geometric deep learning: going beyond euclidean data. *IEEE Signal Processing Magazine*, 34(4): 18–42, 2017.

Bronstein, M. M., Bruna, J., Cohen, T., and Veličković, P. Geometric deep learning: Grids, groups, graphs, geodesics, and gauges. *arXiv preprint arXiv:2104.13478*, 2021.

Cavallari, S., Zheng, V. W., Cai, H., Chang, K. C.-C., and Cambria, E. Learning community embedding with community detection and node embedding on graphs. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pp. 377–386, 2017.

Chen, Z., Li, L., and Bruna, J. Supervised community detection with line graph neural networks. In *International Conference on Learning Representations*, 2018.

Coward, S., Visse-Martindale, E., and Ramesh, C. Attention-based clustering: Learning a kernel from context. *arXiv preprint arXiv:2010.01040*, 2020.

Du, K.-L. Clustering: A neural network approach. *Neural networks*, 23(1):89–107, 2010.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Finley, T. and Joachims, T. Supervised clustering with support vector machines. In *Proceedings of the 22nd international conference on Machine learning*, pp. 217–224, 2005.

Fortunato, S. Community detection in graphs. *Physics reports*, 486(3-5):75–174, 2010.

Gershman, S. and Goodman, N. Amortized inference in probabilistic reasoning. In *Proceedings of the annual meeting of the cognitive science society*, volume 36, 2014.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On Calibration of Modern Neural Networks. In *International Conference on Machine Learning*, pp. 1321–1330, 2017.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30*, pp. 1024–1034, 2017.

Hamilton, W. L. Graph representation learning. *Synthesis Lectures on Artificial Intelligence and Machine Learning*, 14(3):1–159, 2020.

Holland, P. W., Laskey, K. B., and Leinhardt, S. Stochastic blockmodels: First steps. *Social networks*, 5(2):109–137, 1983.

Hubert, L. and Arabie, P. Comparing Partitions. *Journal of Classification*, 1985.

Ienco, D. and Interdonato, R. Deep multivariate time series embedding clustering via attentive-gated autoencoder. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pp. 318–329. Springer, 2020.

Jin, D., Li, B., Jiao, P., He, D., and Shan, H. Community detection via joint graph convolutional network embedding in attribute network. In *International Conference on Artificial Neural Networks*, pp. 594–606. Springer, 2019.

Jin, D., Yu, Z., Jiao, P., Pan, S., Yu, P. S., and Zhang, W. A survey of community detection approaches: From statistical modeling to deep learning. *arXiv preprint arXiv:2101.01669*, 2021.

Kalra, S., Adnan, M., Taylor, G., and Tizhoosh, H. R. Learning permutation invariant representations using memory networks. *European Conference on Computer Vision*, pp. 677–693, 2020.

Kemp, C., Tenenbaum, J. B., Griffiths, T. L., Yamada, T., and Ueda, N. Learning systems of concepts with an infinite relational model. In *AAAI*, volume 3, pp. 5, 2006.

Le, T. A., Baydin, A. G., and Wood, F. Inference compilation and universal probabilistic programming. *Artificial Intelligence and Statistics*, pp. 1338–1348, 2017.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, 2019a.

Lee, J., Lee, Y., and Teh, Y. W. Deep Amortized Clustering. *arXiv:1909.13433*, 2019b.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

Liu, F., Xue, S., Wu, J., Zhou, C., Hu, W., Paris, C., Nepal, S., Yang, J., and Yu, P. S. Deep learning for community detection: Progress, challenges and opportunities. *IJCAI*, 2020.

McLachlan, G. J. and Basford, K. E. *Mixture models: Inference and applications to clustering*, volume 84. Marcel Dekker, 1988.

Miller, J. W. and Harrison, M. T. Mixture models with a prior on the number of components. *Journal of the American Statistical Association*, 113(521):340–356, 2018.

Min, E., Guo, X., Liu, Q., Zhang, G., Cui, J., and Long, J. A survey of clustering with deep learning: From the perspective of network architecture. *IEEE Access*, 6: 39501–39514, 2018.

Mossel, E., Neeman, J., and Sly, A. Consistency thresholds for binary symmetric block models. *arXiv preprint arXiv:1407.1591*, 3(5), 2014.

Pakman, A., Wang, Y., Mitelut, C., Lee, J., and Paninski, L. Neural Clustering Processes. In *International Conference on Machine Learning*, 2020.

Rodriguez, A. and Mueller, P. Nonparametric Bayesian Inference. *NSF-CBMS Regional Conference Series in Probability and Statistics*, 9:i–110, 2013.

Sun, F.-Y., Qu, M., Hoffmann, J., Huang, C.-W., and Tang, J. vgraph: A generative model for joint community detection and node representation learning. *NeurIPS*, 2019.

Tsitsulin, A., Palowitch, J., Perozzi, B., and Müller, E. Graph clustering with graph neural networks. *arXiv preprint arXiv:2006.16904*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Vinh, N. X., Epps, J., and Bailey, J. Information theoretic measures for clusterings comparison: Variants, properties, normalization and correction for chance. *Journal of Machine Learning Research*, 11(Oct):2837–2854, 2010.

Wang, X., Cui, P., Wang, J., Pei, J., Zhu, W., and Yang, S. Community preserving network embedding. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2017.

Xu, Z., Tresp, V., Yu, K., and Kriegel, H.-P. Learning infinite hidden relational models. *Uncertainity in Artificial Intelligence*, 2006.

Yang, J. and Leskovec, J. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.

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

Zaheer, M., Kottur, S., Ravanbakhsh, S., Póczos, B., Salakhutdinov, R., and Smola, A. J. Deep sets. In *Advances in neural information processing systems*, 2017.

A. Experimental Details

A.1. General SBM Dataset

The SBM dataset with $K = 1 \sim 16$ communities are created according to the generative model in Section 5.1. Communities with less than 5 nodes are removed from the resulting graphs. The train, validation and test sets contain 20000, 1000, and 1000 graphs, respectively.

A.2. SNAP Datasets

The SNAP dataset (Leskovec & Krevl, 2014) is distributed under the BSD license, which means that it is free for both academic and commercial use. In each SNAP dataset of real-world graphs, we use the top 5000 high quality communities chosen according to an average of six scoring functions in (Yang & Leskovec, 2015). We randomly split these into 3000 train, 500 validation and 1500 test communities, and extracted subgraphs composed of multiple non-overlapping communities from each split to form the train, validation and test sets. For the 3-community experiments, we find triplets of communities C_1, C_2, C_3 such that they form a connected graph, and that no nodes belong to multiple communities. We filtered for graph size and community imbalance to ensure each pair of communities satisfies $20 < |C_1 \cup C_2| < 500$, $|C_1| < 20|C_2|$, and $|C_2| < 20|C_1|$. For the experiments with $2 \sim 4$ SNAP communities, we first created a community graph in which each node represents a community, and an edge exists between a pair of nodes if the two corresponding communities are not overlapping and satisfy the size/imbalance constraints above. We then extracted subgraphs from the each SNAP dataset by finding cliques of size 2-4 in the community graph. The dataset statistics is shown in Table 5. For datasets with more than 1000 graphs in the test set, the test set is randomly subsetted to 1000 graphs for faster evaluation.

Dataset	Train/Val/Test Networks	$\|V\|$	$\|E\|$
DBLP ($K = 3$)	3929/28/696	65	406
DBLP ($K = 2 \sim 4$)	3222/166/1000	100	542
Youtube ($K = 3$)	8670/1683/1000	100	442
Youtube ($K = 2 \sim 4$)	22141/2687/1000	122	559

Table 5: SNAP dataset statistics.

A.3. Model Training and Inference

All proposed models are implemented in PyTorch and trained with a batch size of 16. The number of training iterations is 5000 for the 3-community SNAP dataset, and 10000 for the SNAP dataset with $2 \sim 4$ communities and the general SBM dataset. The learning rate is 0.0001 for GCN-CCP, GCN-DAC, GCN-CCP-Attn, and 0.00005 for GCN-NCP and GCN-NCP-Attn.

For DMoN (Tsitsulin et al., 2020)[1] and LGNN/GNN (Chen et al., 2018)[2], we used the official implementations with default parameters unless otherwise noted. The DMoN model has a hidden dimension of 512 and is optimized for 1000 iterations. The LGNN/GNN models are trained for the same amount of iterations as our proposed methods on SNAP datasets, and twice amount of iterations on the SBM dataset.

All run time measurements are made on Nvidia P100 GPUs.

B. Network Architectures

B.1. Graph Convolutional Networks (GCN)

GCN is a class of message-passing GNN that updates the representation of each node based on local neighborhood information.

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

[2]LGNN/GNN: https://github.com/zhengdao-chen/GNN4CD

GraphSAGE. GraphSAGE (Hamilton et al., 2017) defines a graph convolution operation that updates the features of each node by integrating the features of both the center and neighboring nodes:

$$h_i^{\ell+1} = \mathrm{ReLU}\Big(U^\ell h_i^\ell + V^\ell \mathrm{Aggregate}_{j\in N_i}\{h_j^\ell\}\Big), \tag{15}$$

where h_i^ℓ is the feature of node i at layer ℓ, N_i is the neighborhood of node i, U^ℓ and V^ℓ are learnable weight matrices of the neural network. The neighborhood aggregation function can be a simple mean function, or more complex LSTM and pooling aggregators. GraphSAGE belongs to isotropic GCNs in which each neighbor node contributes equally to the update function. We use a 4-layer GraphSAGE GCN with the mean aggregator and batch normalization (BN) for our experiments:

$$h_i^{\ell+1} = \mathrm{ReLU}\Big(\mathrm{BN}(U^\ell h_i^\ell + V^\ell \mathrm{Mean}_{j\in N_i}\{h_j^\ell\})\Big). \tag{16}$$

GatedGCN. GatedGCN (Bresson & Laurent, 2018) is an anisotropic GCN that leverages edge gating mechanisms. Each neighboring node in the graph convolution operation may receive different weights depending on the edge gate. Residual connections are used between layers for multi-layer GatedGCN. To improve GatedGCN, (Dwivedi et al., 2020) proposed explicitly updating edge gates across layers:

$$h_i^{\ell+1} = h_i^\ell + \mathrm{ReLU}\Big(\mathrm{BN}(U^\ell h_i^\ell + \sum_{j\to i} e_{ij}^\ell \odot V^\ell h_j^\ell)\Big), \tag{17}$$

where $h_i^0 = x_i, e_{ij}^0 = 1$, and e_{ij}^ℓ is the edge gate computed as follows:

$$e_{ij}^\ell = \frac{\sigma(\hat{e}_{ij}^\ell)}{\sum_{j'\to i}\sigma(\hat{e}_{ij'}^\ell)+\epsilon}, \tag{18}$$

$$\hat{e}_{ij}^\ell = \hat{e}_{ij}^{\ell-1} + \mathrm{ReLU}\Big(\mathrm{BN}(A^\ell h_i^{\ell-1} + B^\ell h_j^{\ell-1} + C^\ell \hat{e}_{ij}^{\ell-1})\Big). \tag{19}$$

We used a 4-layer GatedGCN encoder with hidden dimension of 128 in each layer.

B.2. Attention Modules

Our attention modules are composed of standard Multi-Head Attention (MHA) blocks (Vaswani et al., 2017) , which take as inputs n query vectors $\mathbf{q} = (q_1 \dots q_n)^\top \in \mathbb{R}^{n\times d_q}$, m key vectors $\mathbf{k} = (k_1 \dots k_m)^\top \in \mathbb{R}^{m\times d_k}$, and m value vectors $\mathbf{v} = (v_1 \dots v_m)^\top \in \mathbb{R}^{m\times d_v}$, and return n vectors $\mathrm{MHA}(\mathbf{q},\mathbf{k},\mathbf{v}) \in \mathbb{R}^{n\times dh}$. Using the MHA, we follow (Lee et al., 2019a) in defining three attention modules for functions defined over sets:

- **Multihead Attention Block (MAB).** Given two sets $\mathbf{x} = (x_1,\dots,x_n)^\top \in \mathbb{R}^{n\times d_x}$ and $\mathbf{y} = (y_1,\dots,y_m)^\top \in \mathbb{R}^{m\times d_y}$, we define

$$\mathrm{MAB}(\mathbf{x},\mathbf{y}) = \mathbf{h} + \mathrm{FF}(\mathbf{h}) \in \mathbb{R}^{n\times d} \tag{20}$$

$$\text{where } \mathbf{h} = \mathbf{x}W_q^{[h]} + \mathrm{MHA}(\mathbf{x},\mathbf{y},\mathbf{y}) \in \mathbb{R}^{n\times d}, \tag{21}$$

  with $W_q^{[h]} = [W_q^1 \dots W_q^h] \in \mathbb{R}^{d_x\times d}$ and FF is a feed-forward layer applied to each row of $\mathbf{h}$. Note that in (21), $\mathbf{y}$ is both the key and value set and that FF and MHA have their own trainable parameters.

- **Pooling by Multihead Attention (PMA).** Given a set $\mathbf{x}$, we create a permutation invariant summary into a small number of m vectors with

$$\mathrm{PMA}_m(\mathbf{x}) = \mathrm{MAB}(\mathbf{e},\mathbf{x}) \in \mathbb{R}^{m\times d}$$

  where $\mathbf{e} = (e_1,\dots,e_m) \in \mathbb{R}^{m\times d}$ are trainable parameters. This is a weighted pooling of the items in $\mathbf{x}$, where an attention mechanism determines the weights.

- **Induced Self-Attention Block (ISAB).** The time complexity of expressing self-interactions in a set via $\mathrm{MAB}(\mathbf{x},\mathbf{x})$ scales as $O(n^2)$. To reduce this cost we approximate the full pairwise comparison via a smaller trainable set of inducing points $\mathbf{s} = (s_1,\dots,s_m)$,

$$\mathrm{ISAB}(\mathbf{x}) = \mathrm{MAB}(\mathbf{x},\mathrm{MAB}(\mathbf{s},\mathbf{x})) \in \mathbb{R}^{n\times d}$$

  Thus we indirectly compare the pairs in $\mathbf{x}$ using $\mathbf{s}$ as a bottleneck, with time complexity $O(nm)$. Since $\mathrm{PMA}_m(\mathbf{x})$ is invariant under permutations of $\mathbf{x}$'s rows, ISAB is permutation-equivariant. Higher-order interactions are obtained by stacking multiple ISAB layers.

B.3. CCP Architecture

The CCP model is implemented as described in (Pakman et al., 2020). The full CCP architecture, including the prior, likelihood and posterior components, is illustrated in Figure 9. The posterior network is only used in training, thus not displayed in the diagram of Figure 2 in the main text.

B.3.1. ENCODINGS

In order to parametrize the prior, likelihood and posterior of the CCP model, it is convenient to define first some symmetric encodings for different subsets of the dataset $\mathbf{x}$ at iteration k.

In the main text we used x_a to refer to the first element in cluster $\mathbf{s}_k$, and x_{q_i} to indicate the additional points available to join $\mathbf{s}_k$. Here we use instead x_{d_k}, x_{a_i} respectively, in order to be consistent with the notation of (Pakman et al., 2020). The notation $\mathbf{x}_k$ indicates that the dataset is split into three groups, $\mathbf{x}_k = (\mathbf{x_a}, x_{d_k}, \mathbf{x_s})$, where

$\mathbf{x_a}= (x_{a_1} \dots x_{a_{m_k}})$	m_k available points for cluster k
x_{d_k}	First data point in cluster k
$\mathbf{x_s} = (\mathbf{x}_{\mathbf{s}_1} \dots \mathbf{x}_{\mathbf{s}_{k-1}})$	Points already assigned to clusters.

Let us also define

$$\bar{\mathbf{u}}_\mathbf{a} = (\bar{u}_1 \dots \bar{u}_{m_k}) = u(x_1) \dots u(x_{m_k}) \tag{22}$$

The encodings we need are:

	Definition	Encoded Points
D_k	$= x_{d_k}$	x_{d_k}, the first point in cluster k
U_k	$= \text{mean}(\bar{\mathbf{u}}_\mathbf{a})$	$\mathbf{x_a}$, all the m_k points available to join x_{d_k}
U_k^{in}	$= \text{mean}\Big(\bar{u}_{a_i}, i \in (1 \dots m_k), b_i = 1\Big)$	Points from $\mathbf{x_a}$ that join cluster k.
U_k^{out}	$= \text{mean}\Big(\bar{u}_{a_i}, i \in (1 \dots m_k), b_i = 0\Big)$	Points from $\mathbf{x_a}$ that do not join cluster k
G_k	$= \sum_{j=1}^{k-1} g\Big(\text{mean}\Big(h(x_i), i \in \mathbf{s}_j\Big)$	All the clusters $\mathbf{s}_{1:k-1}$.

(23)

B.3.2. PRIOR AND LIKELIHOOD

Having generated $k-1$ clusters $\mathbf{s}_{1:k-1}$, the elements of $\mathbf{s}_k$ are generated in a process with latent variables $d_k, \mathbf{z}_k$ and joint distribution.

$$p_\theta(\mathbf{s}_k, \mathbf{z}_k, d_k | \mathbf{s}_{1:k-1}, \mathbf{x}) = p_\theta(\mathbf{b}_k | \mathbf{z}_k, \mathbf{x}_k) p_\theta(\mathbf{z}_k | \mathbf{x}_k) p(d_k | \mathbf{s}_{1:k-1}) \,, \tag{24}$$

where

$$p_\theta(\mathbf{b}_k | \mathbf{z}_k, \mathbf{x}_k) = \prod_{i=1}^{m_k} \varphi(b_i | \mathbf{z}_k, U, G, x_{d_k}, x_{a_i}) \,. \tag{25}$$

The priors and likelihood are

$$p(d_k | \mathbf{s}_{1:k-1}) = \begin{cases} 1/|I_k| & \text{for } d_k \in I_k \,, \\ 0 & \text{for } d_k \notin I_k \,, \end{cases} \tag{26}$$

$$p_\theta(\mathbf{z}_k | \mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k | \mu(\mathbf{x}_k), \sigma(\mathbf{x}_k)) \tag{27}$$

$$\varphi(b_i | \mathbf{z}_k, U, G, x_{d_k}, x_{a_i}) = \text{sigmoid}[\rho_i(\mathbf{z}_k, \mathbf{x}_k)] \tag{28}$$

where I_k is the set of indices available to become the first element of $\mathbf{s}_k$, and we have defined

$$\begin{aligned}
\mu(\mathbf{x}_k) &= \mu(D_k, U_k, G_k) && (29)\\
\sigma(\mathbf{x}_k) &= \sigma(D_k, U_k, G_k), && (30)\\
\rho_i(\mathbf{z}_k, \mathbf{x}_k) &= \rho(\mathbf{z}_k, x_{a_i}, D_k, U_k, G_k) \qquad i = 1 \dots m_k && (31)
\end{aligned}$$

where μ, σ, ρ are represented with MLPs. Note that in all the cases the functions depend on encodings in (23) that are consistent with the permutation symmetries dictated by the conditioning information.

B.3.3. POSTERIOR

To learn the prior and likelihood functions, we introduce

$$q_\phi(\mathbf{z}_k, d_k|\mathbf{s}_{1:k}, \mathbf{x}) = q_\phi(\mathbf{z}_k|\mathbf{b}_k, d_k, \mathbf{x}_k) q_\phi(d_k|\mathbf{s}_{1:k}, \mathbf{x}) \tag{32}$$

to approximate the intractable posterior. This allows us to train CCP as a conditional variational autoencoder (VAE) (Sohn et al., 2015).

For the first factor we assume a form

$$q_\phi(\mathbf{z}_k|\mathbf{b}_k, d_k, \mathbf{x}_k) = \mathcal{N}(\mathbf{z}_k|\mu_q(D_k, A_k^{in}, A_k^{out}, G_k), \sigma_q(D_k, A_k^{in}, A_k^{out}, G_k)) \tag{33}$$

where μ_q, σ_q are MLPs. For the second factor we assume

$$q(d_k|\mathbf{s}_{1:k}) = \begin{cases} 1/N_k & \text{for } d_k \in \mathbf{s}_k, \\ 0 & \text{for } d_k \notin \mathbf{s}_k. \end{cases} \tag{34}$$

This approximation is very good in cases of well separated clusters. Since $q(d_k|\mathbf{s}_{1:k})$ has no parameters, this avoids the problem of backpropagation through discrete variables.

B.3.4. ELBO

The ELBO that we want to maximize is given by

$$\begin{aligned}
&\mathbb{E}_{p(\mathbf{x}, \mathbf{s}_{1:K})} \log p_\theta(\mathbf{s}_{1:K}|\mathbf{x}) && (35)\\
&= \mathbb{E}_{p(\mathbf{x}, \mathbf{s}_{1:K})} \sum_{k=1}^{K} \log \left[\sum_{d_k=1}^{N_k} \int d\mathbf{z}_k p_\theta(\mathbf{s}_k, \mathbf{z}_k, d_k|\mathbf{s}_{1:k-1}, \mathbf{x}) \right] && (36)\\
&\geq \mathbb{E}_{p(\mathbf{x}, \mathbf{s}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_k, d_k|\mathbf{s}_{1:k}, \mathbf{x})} \log \left[\frac{p_\theta(\mathbf{s}_k, \mathbf{z}_k, d_k|\mathbf{s}_{1:k-1}, \mathbf{x})}{q_\phi(\mathbf{z}_k, d_k|\mathbf{s}_{1:k}, \mathbf{x})} \right] && (37)\\
&= \mathbb{E}_{p(\mathbf{x}, \mathbf{s}_{1:K})} \sum_{k=1}^{K} \mathbb{E}_{q_\phi(\mathbf{z}_k, d_k|\mathbf{s}_{1:k}, \mathbf{x})} \log \left[\frac{p_\theta(\mathbf{b}_k|\mathbf{z}_k, \mathbf{x}_k) p_\theta(\mathbf{z}_k|\mathbf{x}_k) p(d_k|\mathbf{s}_{1:k-1})}{q_\phi(\mathbf{z}_k|\mathbf{b}_k, d_k, \mathbf{x}_k) q_\phi(d_k|\mathbf{s}_{1:k}, \mathbf{x})} \right] && (38)
\end{aligned}$$

B.4. CCP-Attn Architecture

In CCP-Attn, we replace equation (22) with a ISAB self attention layer among all available points:

$$\begin{aligned}
(\bar{u}_{d_k}, \bar{u}_1 \dots \bar{u}_{m_k}) &= \text{ISAB}[u(x_{d_k}), u(x_1) \dots u(x_{m_k})] && (39)\\
\bar{\mathbf{u}}_\mathbf{a} &= (\bar{u}_1 \dots \bar{u}_{m_k})
\end{aligned}$$

and replace the encodings in (23) with attention-based aggregations:

Together, these changes give rise to the CCP-Attn architecture illustrated in Figure 9.

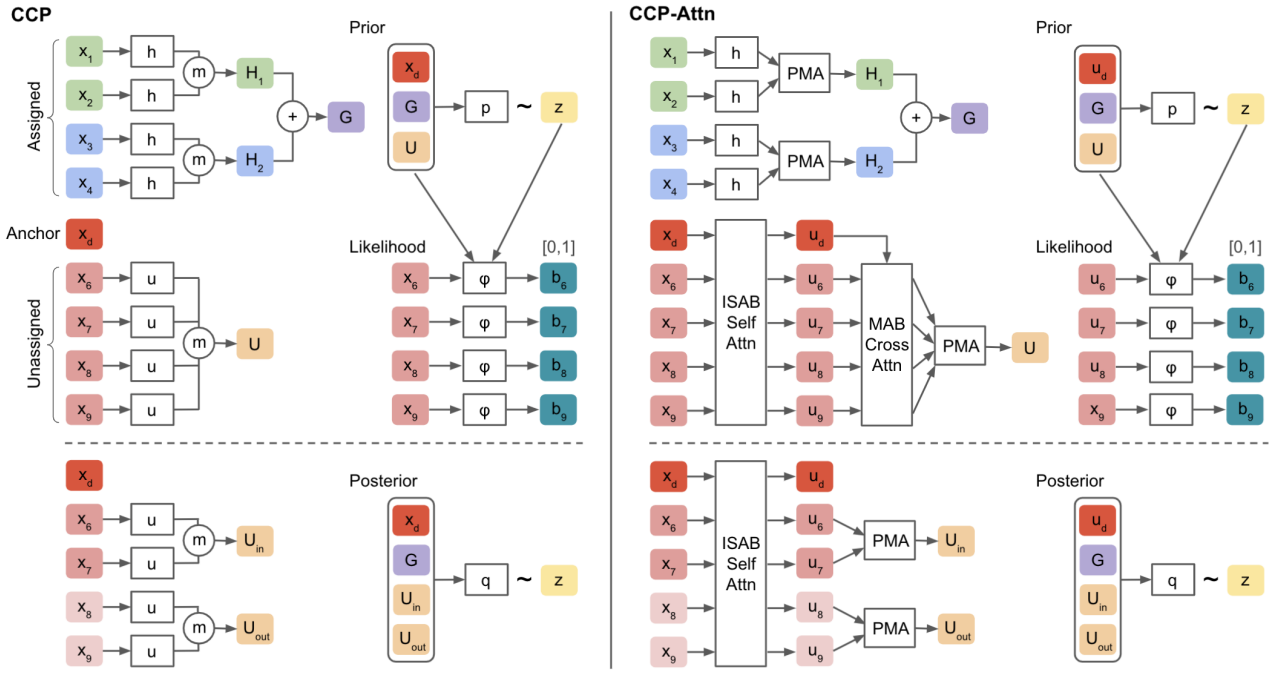

Figure 9: **Full architecture of CCP and CCP-Attn.** The diagram illustrates the conditional prior, likelihood and posterior components of CCP and CCP-Attn. The mean aggregations ⓜ used by CCP (see equation are replaced in CCP-Attn by Set Transformer attention modules from (Lee et al., 2019a).

	Definition	Encoded Points
D_k	$= \bar{u}_{d_k}$	x_{d_k}, the first point in cluster k
U_k	$= \mathrm{PMA}(\mathrm{MAB}(\bar{\mathbf{u}}_\mathbf{a}, \bar{u}_d))$	$\mathbf{x_a}$, all the m_k points available to join x_{d_k}
A_k^{in}	$= \mathrm{PMA}\Big(\bar{u}_{a_i}, i \in (1 \dots m_k), b_i = 1\Big)$	Points from $\mathbf{x_a}$ that join cluster k.
A_k^{out}	$= \mathrm{PMA}\Big(\bar{u}_{a_i}, i \in (1 \dots m_k), b_i = 0\Big)$	Points from $\mathbf{x_a}$ that do not join cluster k
G_k	$= \sum_{j=1}^{k-1} g\Big(\mathrm{PMA}\Big(h(x_i), i \in \mathbf{s}_j\Big)$	All the clusters $\mathbf{s}_{1:k-1}$.

(40)

B.4.1. Neural Networks in CCP and CCP-Attn

The h, g and u functions are parameterized by 3-layer MLPs. The p, q and φ functions are MLPs with 5, 5, and 4 layers, respectively. All MLPs have hidden-layer dimensions of 128 and parametric ReLU (PReLU) layers in between linear layers. Each vector in the equations above has a dimension of 128. In CCP-Attn, the ISAB, MAB, and PMA attention modules contain 32 inducing points, 4 attention heads, and hidden-layer dimensions of 128.

B.5. NCP and NCP-Attn Architecture

The NCP architecture is described in (Pakman et al., 2020). In NCP, each cluster k is encoded by permutation-invariant aggregation of data points assigned to that cluster

$$H_k = \sum_{i:c_i=k} h(x_i) \quad h : \mathbb{R}^{d_x} \to \mathbb{R}^{d_h}. \tag{41}$$

The global representation of the current clustering configuration is given by

$$G = \sum_{k=1}^{K} g(H_k), \quad g : \mathbb{R}^{d_h} \to \mathbb{R}^{d_g}. \tag{42}$$

Given $n-1$ assigned points $x_{1:n-1}$ and their cluster labels $c_{1:n-1}$, we want to find the cluster label for the next point x_n. At this point, the unassigned points $x_{n+1:N}$ are represented by

$$U = \sum_{i=n+1}^{N} u(x_i), \qquad u : \mathbb{R}^{d_x} \to \mathbb{R}^{d_u}. \tag{43}$$

The probability of the next point x_n joining cluster k is modeled by the variable-input softmax function

$$q_\theta(c_n = k | c_{1:n-1}, \mathbf{x}) = \frac{e^{f(G_k, U)}}{\sum_{k'=1}^{K+1} e^{f(G_{k'}, U)}}. \tag{44}$$

The h, u, g, f functions in NCP are MLPs with 2, 2, 5, and 5 layers, respectively. The MLPs have hidden-layer dimensions of 128 and parametric ReLU (PReLU) layers in between linear layers. In NCP-Attn, the h and u functions are replaced by ISAB attention across all points. The ISAB attention module contains 32 inducing points, 4 attention heads, and hidden-layer dimensions of 128.

B.6. DAC Architecture

The DAC model is composed of the same neural network backbone as CCP-Attn, and the binary cross entropy loss function of the Anchored Filtering method in (Lee et al., 2019b).

C. Additional Experiments

Figure 10 illustrates how our Bayesian approach allows quantifying the uncertainty in the number of clusters for a range of parameters in this model.

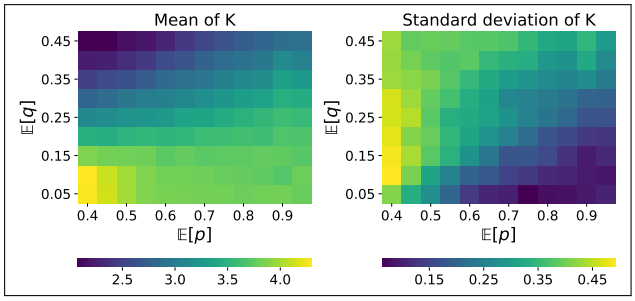

Figure 10: **Quantifying uncertainty of inference on General SBM.** Mean (*left*) and standard deviation (*right*) of the inferred number of clusters K across 500 posterior samples. The model is trained on SBM data generated from $N \sim \text{Unif}[50, 300]$, $c_1 \dots c_N \sim \text{CRP}(0.7)$, $p = \phi_{k_1,k_2} \sim \text{Beta}(6, 3)$ for $k_1 = k_2$, and $q = \phi_{k_1,k_2} \sim \text{Beta}(1, 5)$ for $k_1 \neq k_2$. The inference is run on SBM graphs ($N = 200$, $K = 4$, equal partition) with varying connection probabilities $p = \phi_{k_1=k_2}$ and $q = \phi_{k_1 \neq k_2}$ generated from $\phi_{k_1,k_2} \sim \text{Beta}(\alpha, \beta)$ with $\alpha + \beta = 10$. The results are averaged over 100 test examples. As expected, the standard-deviation is takes minimum values for large mean p and small mean q.

Appendix References

Bresson, X. and Laurent, T. Residual gated graph convnets. *arXiv preprint arXiv:1711.07553*, 2018.

Chen, Z., Li, L., and Bruna, J. Supervised community detection with line graph neural networks. In *International Conference on Learning Representations*, 2018.

Dwivedi, V. P., Joshi, C. K., Laurent, T., Bengio, Y., and Bresson, X. Benchmarking graph neural networks. *arXiv preprint arXiv:2003.00982*, 2020.

Hamilton, W., Ying, Z., and Leskovec, J. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems 30*, pp. 1024–1034, 2017.

Lee, J., Lee, Y., Kim, J., Kosiorek, A., Choi, S., and Teh, Y. W. Set transformer: A framework for attention-based permutation-invariant neural networks. In *International Conference on Machine Learning*, 2019a.

Lee, J., Lee, Y., and Teh, Y. W. Deep Amortized Clustering. *arXiv:1909.13433*, 2019b.

Leskovec, J. and Krevl, A. SNAP Datasets: Stanford large network dataset collection. http://snap.stanford.edu/data, June 2014.

Pakman, A., Wang, Y., Mitelut, C., Lee, J., and Paninski, L. Neural Clustering Processes. In *International Conference on Machine Learning*, 2020.

Sohn, K., Lee, H., and Yan, X. Learning structured output representation using deep conditional generative models. In *Advances in neural information processing systems*, pp. 3483–3491, 2015.

Tsitsulin, A., Palowitch, J., Perozzi, B., and Müller, E. Graph clustering with graph neural networks. *arXiv preprint arXiv:2006.16904*, 2020.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. Attention is all you need. In *Advances in Neural Information Processing Systems*, 2017.

Yang, J. and Leskovec, J. Defining and evaluating network communities based on ground-truth. *Knowledge and Information Systems*, 42(1):181–213, 2015.