Unsupervised Joint k-node Graph Representations with Compositional Energy-Based Models

Leonardo Cotta[*]
Purdue University
cotta@purdue.edu

Carlos H. C. Teixeira
Universidade Federal de Minas Gerais, Brazil
carlos@dcc.ufmg.br

Ananthram Swami
United States Army Research Laboratory
ananthram.swami.civ@mail.mil

Bruno Ribeiro
Purdue University
ribeiro@cs.purdue.edu

Abstract

Existing Graph Neural Network (GNN) methods that learn *inductive unsupervised* graph representations focus on learning node and edge representations by predicting observed edges in the graph. Although such approaches have shown advances in downstream node classification tasks, they are ineffective in jointly representing larger k-node sets, $k>2$. We propose MHM-GNN, an inductive unsupervised graph representation approach that combines joint k-node representations with energy-based models (hypergraph Markov networks) and GNNs. To address the intractability of the loss that arises from this combination, we endow our optimization with a loss upper bound using a finite-sample unbiased Markov Chain Monte Carlo estimator. Our experiments show that the unsupervised joint k-node representations of MHM-GNN produce better unsupervised representations than existing approaches from the literature.

1 Introduction

Inductive unsupervised learning using Graph Neural Networks (GNNs) in (dyadic) graphs is currently restricted to node and edge representations due to their reliance on edge-based losses [8, 21, 29, 64]. If we want to tackle downstream tasks that require jointly reasoning about $k > 2$ nodes, but whose input data are dyadic relations (i.e., standard graphs) rather than hyperedges, we must develop techniques that can go beyond edge-based losses.

Joint k-node representation tasks with dyadic relational inputs include drone swarms that communicate amongst themselves to jointly act on a task [56, 59], but also include more traditional product-recommendation tasks. For instance, an e-commerce website might want to predict which k products could be jointly purchased in the same shopping cart, while the database only records (product, product) dyads to safeguard user information.

Srinivasan and Ribeiro [57] have recently shown that GNN node representations are insufficient to capture joint characteristics of k nodes that are unique to this group of nodes. Indeed, our experiments show that using existing unsupervised GNN —with their node representations and edge losses— one cannot accurately detect these k-product carts on an e-commerce website. Unfortunately, existing GNN extensions that give joint k-node representations require *supervised* graph-wide losses [39, 36], leaving a significant gap between edge and *supervised* whole-graph losses (i.e., we need multiple labeled graphs for these to work). The main reason for this gap is scalability: to obtain *true*

[*] http://cottascience.github.io/

Preprint. Under review.

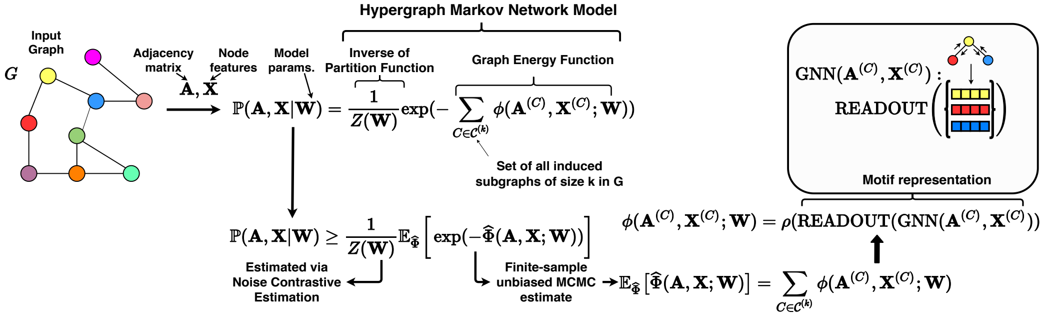

Hypergraph Markov Network Model
Input Graph
G
Adjacency matrix
Node features
Model params.
$\mathbf{A}, \mathbf{X}$
Inverse of Partition Function
Graph Energy Function
$\mathbb{P}(\mathbf{A}, \mathbf{X} | \mathbf{W}) = \frac{1}{Z(\mathbf{W})} \exp(-\sum_{C \in \mathcal{C}^{(k)}} \phi(\mathbf{A}^{(C)}, \mathbf{X}^{(C)}; \mathbf{W}))$
Set of all induced subgraphs of size k in G
$\mathrm{GNN}(\mathbf{A}^{(C)}, \mathbf{X}^{(C)})$:
READOUT
Motif representation
$\phi(\mathbf{A}^{(C)}, \mathbf{X}^{(C)}; \mathbf{W}) = \rho(\mathrm{READOUT}(\mathrm{GNN}(\mathbf{A}^{(C)}, \mathbf{X}^{(C)}))$
$\mathbb{P}(\mathbf{A}, \mathbf{X} | \mathbf{W}) \geq \frac{1}{Z(\mathbf{W})} \mathbb{E}_{\widehat{\Phi}} \left[\exp(-\widehat{\Phi}(\mathbf{A}, \mathbf{X}; \mathbf{W})) \right]$
Estimated via Noise Contrastive Estimation
Finite-sample unbiased MCMC estimate
$\mathbb{E}_{\widehat{\Phi}} [\widehat{\Phi}(\mathbf{A}, \mathbf{X}; \mathbf{W})] = \sum_{C \in \mathcal{C}^{(k)}} \phi(\mathbf{A}^{(C)}, \mathbf{X}^{(C)}; \mathbf{W})$

Figure 1: The proposed unsupervised graph representation using motif compositions. Here, we present the MHM-GNN model from Equation (1), the energy estimator $\widehat{\Phi}$ from Equation (4), the motif energy and representation from Equation (2).

unsupervised joint k-node representations, one must optimize a model defined over *all* k-node induced subgraphs of a graph.

Our approach MHM-GNN (Motif Hypergraph Markov Graph Neural Networks) leverages the compositionality of hypergraph Markov network models (HMNs) [53, 74, 31] that allows us to define an unsupervised objective (energy-based model) over GNN representations of motifs (see upper half of Figure 1).

Scalability is the main challenge we have to overcome, a type of scalability issue not addressed in the hypergraph Markov network literature [53, 74, 31]. First, there is the traditional likelihood intractability associated with computing the partition function $Z(\mathbf{W})$ of energy models —$Z(\mathbf{W})$ is shown in the likelihood $\mathbb{P}(\boldsymbol{A}, \boldsymbol{X} | \mathbf{W})$ in Figure 1 and also in Equation (1). There are standard solutions for this challenge (e.g., Noise-Contrastive Estimation (NCE) [20]). The more vexing challenge comes from the intractability created by our inductive graph representation that applies motif representations to all k-node subgraphs, which requires $\binom{n}{k}$ operations per gradient step, typically with $n \gg k$. To make this step tractable, we leverage recent advances in finite-sample unbiased Markov Chain Monte Carlo estimation for sums of subgraph functions over large graphs [61]. This unbiased estimate, combined with Jensen's inequality, allows us to optimize a lower bound on the intractable likelihood (assuming $Z(\mathbf{W})$ is known). Fold that into the asymptotics of NCE and we get a principled, tractable optimization.

Contributions. Our contributions are three-fold. First, we introduce MHM-GNN, which produces joint $(k > 2)$-node representations, where k is a hyperparameter of the model. Second, we introduce a principled and scalable stochastic optimization method that learns MHM-GNN with a finite-sample unbiased estimator of the graph energy (see Fig. 1) and a NCE objective. Finally, we show how the joint k-node representations from MHM-GNN produce better unsupervised joint k-node representations than existing approaches that aggregate node representations.

2 Related Work

In this section, we briefly review existing approaches to *inductive unsupervised* representation learning of graphs, discuss existing work with higher-order graph representations and overview energy-based models. Finally, we present what in literature is not related to this work.

Edge-based graph models. Although graph models are prominent in many areas of research [43], most of the proposed models, such as the initial Erdös-Rényi model [15], stochastic block models [25] and the more recent neural network-based approaches [29, 21, 8] assume conditional independence of edges, resulting in what is often called an edge-based loss function. That is, all such models assume the appearance of edges in the graph is independent given the edge representations, which is usually computed via their endpoints' representations. This important conditional independence assumption appears in what we call edge-based graph models. There exist alternatives such as Markov Random Graphs [18], where an edge is dependent on every other edge that shares one of its endpoints, but graph models without any conditional independence assumption are still not commonly used.

Inductive unsupervised node representations with GNNs. Recently, GraphSAGE [21] introduced the use of GNNs to learn inductive node representations in an unsupervised manner by applying an edge-based loss while using short random walks. There are also auto-encoder approaches [29, 45, 54], where one tries to reconstruct the edges in the graph using node representations. Auto-encoders also assume conditional independence of edges and can be classified as edge-based models. In contrast to edge-based loss models, DGI [64] minimizes the mutual entropy between node representations and a whole-graph representation —it does not model a probability distribution. Whilst the combination of GNNs and edge-based models has been shown to be effective in representing nodes and edges, *i.e.* $k = 1$ and $k = 2$ representations, moving to $k > 2$ joint representations requires a model with higher-order factorization. To this end, we introduce MHM-GNN, a model that leverages hypergraph Markov networks and GNNs to generate k-node motif representations.

Joint k-node representations with dyadic graph. Recently, Morris et al. [39] and Maron et al. [36] proposed higher-order neural architectures to represent entire graphs in a supervised learning setting, as opposed to the unsupervised setting discussed in this work. Moreover, we also point how since these higher-order GNN approaches are concerned with representing entire graphs in a supervised setting, the subgraph size k is treated as a constant and scalability is not addressed (models already overfit with small k). Our approach can incorporate higher-order GNNs and also the more recent Relational Pooling framework [40](see Equation (2)). We can summarize previous efforts to represent subgraphs in an unsupervised manner as sums of the individual nodes' representations [22]. Hypergraph neural network models [69, 3, 16] require observing polyadic data, while here we are interested in modeling dyadic data. We provide a broader discussion of higher-order graph models and the challenges of translating supervised approaches to an unsupervised setting in the supplement.

Energy-based models. Energy-Based Models (EBMs) have been widely used to learn representations of images [50], text [4], speech [60] and many other domains. In general, works in EBMs come in two flavors: what model to use for the energy and how to estimate the partition function $Z(\mathbf{W})$, which is usually intractable. For the latter, there are model-specific MCMC methods, such as Contrastive Divergence [23] and standard solutions, such as the one we choose in this work: Noise-Constrastive Estimation (NCE). As for the energy model, we opt for a hypergraph Markov network [53, 74, 31]. The energy of a graph is given by all of its $\binom{n}{k}$ subgraphs, which induces a new kind of intractability in the energy computation. Thus, we propose an unbiased energy estimation procedure in Section 4, which provides an upper bound on our NCE objective.

Unrelated work. It is important to not confuse *learning inductive unsupervised joint k-node* representations and other existing graph representation methods [39, 36, 19, 49, 51, 52, 34, 70]. Although motif-aware methods [34, 52] explicitly use motif information, they are used to build node representations rather than joint k-node representations, and thus, are equatable to other more powerful node representations, such as those in Hamilton et al. [21], Veličković et al. [64]. Here, we are interested in inductive tasks, hence transductive node representations, like Grover and Leskovec [19], Perozzi et al. [49], are unrelated. Nevertheless, as a matter of curiosity, we provide results for transductive node representations in our joint k-node tasks in the supplement, showing that our approach also works well compared to transductive settings even though our approach was not designed for transductive tasks. Supervised higher-order approaches [39, 36] extract whole-graph representations, which cannot be directly translated to existing unsupervised settings (see supplement for more details on these challenges). We are interested in methods that can be used in end-to-end representation learning, thus feature engineering and extraction, such as those used in graph kernels [70] are not of interest. Finally, the large body of work exploring hyperlink prediction in hypergraphs [7, 48, 73, 68, 71, 72] requires observing polyadic data (hypergraphs) and are transductive, as opposed to our work, where we consider observing dyadic data and propose an inductive model.

3 Motif Hypergraph Markov Graph Neural Networks (MHM-GNN)

In this section we start by introducing notation to then briefly introduce hypergraph Markov networks (HMNs), describe MHM-GNN with an HMN model, and discuss possible GNN-based energy functions used to represent motifs.

Notation. The i-th row of a matrix $\boldsymbol{M}$ will be denoted $\boldsymbol{M}_{i\cdot}$, and its j-th column $\boldsymbol{M}_{\cdot j}$. For the sake of simplicity, we will focus on graphs without edge attributes, even though our model can handle them using the GNN formulation from Battaglia et al. [6]. We denote a graph with n nodes by $G = (V, E, \boldsymbol{X})$, where V is the set of nodes, $E \subseteq V^2$ the edge set, $\boldsymbol{A} \in \{0, 1\}^{n \times n}$ its corresponding adjacency matrix and the matrix $\boldsymbol{X} \in \mathbb{R}^{n \times p}$ encodes the p node features of all n nodes. Each set of k nodes from a graph $C \subseteq V : |C| = k$ has an associated induced subgraph $G^{(C)} = (V^{(C)}, E^{(C)}, \boldsymbol{X}^{(C)})$ (see Definition 1). Induced subgraphs are also referred to as *motifs, graphlets, graph fragments or subgraphs*. Here, we will interchangeably refer to them as (induced) subgraphs or motifs.

Definition 1 (Induced Subgraph)**.** *Let $C \subseteq V : |C| = k$ be a set of k nodes from V with corresponding sorted sequence $\overrightarrow{C} = [C_1, ..., C_k] : C_i < C_{i+1}, C_i \in C \, \forall i \in \{1, ..., k\}$. Then, $G^{(C)} = (V^{(C)}, E^{(C)}, \boldsymbol{X}^{(C)})$ is the induced subgraph of C in G, with adjacency matrix $\mathbf{A}^{(C)}$, where $V^{(C)} = \{1, ..., k\}$, $\boldsymbol{A} \in \{0,1\}^{k \times k} : \boldsymbol{A}^{(C)}_{ij} = \boldsymbol{A}_{C_i C_j}$ and $\boldsymbol{X}^{(C)} \in \mathbb{R}^{k \times p} : \boldsymbol{X}^{(C)}_{i\cdot} = \boldsymbol{X}_{C_i \cdot}$.*

Hypergraph Markov Networks (HMNs). A Markov Network (MN) defines a joint probability distribution as a product of non-negative functions (potentials) over maximal cliques of an undirected graphical model [27, 5]. Although defined over maximal cliques, scalable techniques often assume factorization over non-maximal cliques [53, 5, 74], such as Pairwise Markov Networks (PMNs) [24], where the distribution is expressed as a product of edge potentials. In contrast, since we are interested in learning joint representations of k-node subgraphs, we need a hypergraph Markov network (HMN) (Definition 2), which is an MN model that can encompass all the variables of a $(k > 2)$-node subgraph.

Our graph model is an HMN. HMNs are to PMNs what hypergraphs are to graphs. In HMNs, the joint distribution is expressed as a product of potentials of hyperedges rather than edges. Since in HMNs potentials are defined over subsets of random variables of any size, we have the flexibility to do it over k-node subgraphs. There are previous works referring to HMNs as higher-order graphical models [53, 74], however we find the hypergraph analogy more clarifying. Next, we provide a formal definition of HMNs.

Definition 2 (Hypergraph Markov Networks (HMNs))**.** *A hypergraph Markov network is a Markov network where the joint probability distribution of $\mathbf{Y} = \{Y_1, ..., Y_l\}$ can be expressed as $\mathbb{P}(\mathbf{Y} = \mathbf{y}) = \frac{1}{Z} \Pi_{h \in \mathcal{H}} \phi_h(\mathbf{y}_h)$, where Z is the partition function $Z = \sum_{\mathbf{y}' \in \mathbf{Y}} \Pi_{h \in \mathcal{H}} \phi_h(\mathbf{y}'_h)$, $\phi_h(\cdot) \geq 0$ are non-negative, $\mathcal{H} \subseteq \mathcal{P}(\mathbf{Y}) \backslash \{\emptyset\}$, where $\mathcal{P}(\mathbf{Y})$ is the powerset of a set $\mathbf{Y}$, and $\mathcal{H}$ is the set of hyperedges in the Markov network, $\mathbf{Y}_h$ are the random variables associated with hyperedge h and $\mathbf{y}, \mathbf{y}_h$ assignments of $\mathbf{Y}$ and $\mathbf{Y}_h$ respectively. Finally, an energy-based HMN assumes strictly positve potentials, resulting in the model $\mathbb{P}(\mathbf{Y} = \mathbf{y}) = \frac{1}{Z} \Pi_{h \in \mathcal{H}} \exp(-\phi_h(\mathbf{y}_h)) = \frac{1}{Z} \exp(-\sum_{h \in \mathcal{H}} \phi_h(\mathbf{y}_h))$, where $\phi_h(\cdot)$ is called the energy function of h.*

3.1 MHM-GNNs

We model $\mathbb{P}(\mathbf{A}, \boldsymbol{X} | \mathbf{W})$ with an energy-based HMN, as described in Definition 2, where a hyperedge corresponds to an induced subgraph of k nodes in the graph G. More precisely, for every set of $k > 1$ nodes in the graph $C \subseteq V, |C| = k$, we define a hyperedge $h = \{\mathbf{A}_{ij} : (i,j) \in C^2\} \cup \{\boldsymbol{X}_{i,\cdot} : i \in C\}$ in the HMN to encompass every node variable in the k-node set and every edge variable with both endpoints in it. A hyperedge can be indexed by a set of nodes C, since its corresponding set of random variables is given by the features $\boldsymbol{X}^{(C)}$ and the adjacency matrix $\mathbf{A}^{(C)}$ of the subgraph induced by C, following Definition 1. Thus, a graph with n nodes will have an HMN with $\binom{n}{k}$ potentials. We formally define the model in Definition 3.

Definition 3 (MHM-GNN)**.** *Let $\mathcal{C}^{(k)}$ denote the set of all $\binom{n}{k}$ combinations of k nodes from G. We define a hypergraph Markov Network with a set of hyperedges $\{\{\boldsymbol{A}_{ij} : (i,j) \in C\} \cup \{\boldsymbol{X}_{i,\cdot} : i \in C\} \; : \; C \in \mathcal{C}^{(k)}\}$, which following Definitions 1 and 2, entails the model*

$$\mathbb{P}(\mathbf{A}, \boldsymbol{X} | \mathbf{W}) = \frac{\exp\left(-\sum_{C \in \mathcal{C}^{(k)}} \phi(\mathbf{A}^{(C)}, \boldsymbol{X}^{(C)}; \mathbf{W})\right)}{Z(\mathbf{W})}, \quad (1)$$

where $\phi(\cdot, \cdot; \mathbf{W})$ is an energy function with parameters $\mathbf{W}$ and $Z(\mathbf{W})$ is the partition function given by $Z(\mathbf{W}) = \sum_{n=1}^{\infty} \sum_{\mathbf{A}' \in \{0,1\}^{n \times n}} \int_{\boldsymbol{X}' \in \mathbb{R}^{n \times p}} \exp(-\sum_{C \in \mathcal{C}^{(k)}} \phi(\mathbf{A}'^{(C)}, \boldsymbol{X}'^{(C)}; \mathbf{W})) d\boldsymbol{X}'$.

Although MHM-GNN factorizes the total energy of a graph, the model does not assume any conditional independence between edge variables for $k > 3$. For $k = 2$, the model recovers existing edge-based models and for $k = 3$ edge variables are dependent only on edges that share one of their endpoints, recovering the Markov random graphs class [18]. Furthermore, MHM-GNN will learn a jointly exchangeable distribution [44] if the subgraph energy function $\phi(.,.;\mathbf{W})$ is jointly exchangeable, such as a GNN. In the supplement we connect MHM-GNN assumptions, exchangeability and Exponential Random Graph Models (ERGMs).

Subgraph energy function and representations. As mentioned, to have a jointly exchangeable model with MHM-GNN, we need an energy function $\phi(\mathbf{A}^{(C)}, \boldsymbol{X}^{(C)}; \mathbf{W})$ that is jointly exchangeable with respect to the subgraph $G^{(C)}$. To this end, we break down $\phi(\mathbf{A}^{(C)}, \boldsymbol{X}^{(C)}; \mathbf{W})$ into a composition of two functions. First, we compute a jointly exchangeable representation of $G^{(C)}$, then we use it as input to a more general function that assigns an energy value to the subgraph. Following recent GNN advances [14, 67, 39], we define the subgraph representation with a permutation invariant (READOUT) function over the nodes' representations given by a GNN, denoted by $h^{(C)}(\mathbf{A}^{(C)}, \boldsymbol{X}^{(C)}; \mathbf{W}_{\text{GNN}}, \mathbf{W}_{\text{R}}) = \text{READOUT}(\text{GNN}(\mathbf{A}^{(C)}, \boldsymbol{X}^{(C)}; \mathbf{W}_{\text{GNN}}); \mathbf{W}_{\text{R}})$. Usually, the READOUT function is a row-wise sum followed by a multi-layer perceptron. Note that, although we choose a 1-GNN approach to represent the subgraph here, any jointly exchangeable graph representation can be used to represent the subgraph, such as k-GNNs [39] and Relational Pooling [40].

Finally, we can define the energy of a subgraph $G^{(C)}$ as

$$\phi(\mathbf{A}^{(C)}, \boldsymbol{X}^{(C)}; \mathbf{W}) = \mathbf{W}_{\text{energy}}^{T} \rho(h^{(C)}(\mathbf{A}^{(C)}, \boldsymbol{X}^{(C)}; \mathbf{W}_{\text{GNN}}, \mathbf{W}_{\text{R}}); \mathbf{W}_{\rho}) \qquad (2)$$

where the model set of weights is $\mathbf{W} = \{\mathbf{W}_{\text{energy}}, \mathbf{W}_{\text{R}}, \mathbf{W}_{\rho}, \mathbf{W}_{\text{GNN}}\}$, $\rho(\cdot; \mathbf{W}_{\rho})$ is a permutation sensitive function with parameters $\mathbf{W}_{\rho}$ such as a multi-layer perceptron with range in $\mathbb{R}^{1\times H}$ and $\mathbf{W}_{\text{energy}} \in \mathbb{R}^{1\times H}$ is a (learnable) weight matrix.

Although the functional form of the distribution and subgraph representations are properly defined, directly computing both the partition function and the total energy of a graph are computationally intractable for an arbitrary k. Therefore, in the next section we discuss how to properly learn the distribution parameters, providing a principled and scalable approximate method.

4 Learning MHM-GNNs

In this section, we first define our unsupervised objective through Noise-Contrastive Estimation (NCE) and then show how to approximate it.

Noise-Contrastive Estimation (NCE). Since directly computing $Z(\mathbf{W})$ of Equation (1) is intractable, we use Noise-Contrastive Estimation (NCE) [20]. In NCE, the model parameters are learned by contrasting observed data and negative (noise) sampled examples. Given the set $\mathcal{D}_{\text{true}}$ of observed graphs and $M|\mathcal{D}_{\text{true}}|$ sampled noise graphs from a noise distribution $\mathbb{P}_n(\mathbf{A}, \boldsymbol{X})$ composing the set D_{noise}, we can define the loss function to be minimized as

$$\begin{aligned}\mathcal{L}(\mathbf{A}, \boldsymbol{X}; \mathbf{W}) = &- \sum_{\mathbf{A}\in\mathcal{D}_{\text{true}}} \log(\hat{y}(\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W}), \mathbb{P}_n(\mathbf{A}, \boldsymbol{X}))) \\ &- \sum_{\mathbf{A}\in\mathcal{D}_{\text{noise}}} \log(1 - \hat{y}(\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W}), \mathbb{P}_n(\mathbf{A}, \boldsymbol{X}))).\end{aligned}$$

with $\hat{y}(\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W}), \mathbb{P}_n(\mathbf{A}, \boldsymbol{X})) = \sigma(-\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W}) - \log(M\mathbb{P}_n(\mathbf{A}, \boldsymbol{X})))$, where $\sigma(\cdot)$ is the sigmoid function and $\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W}) = \sum_{C\in\mathcal{C}^{(k)}} \phi(\mathbf{A}^{(C)}, \boldsymbol{X}^{(C)}; \mathbf{W})$ denotes the total energy of a graph $G = (V, E, \boldsymbol{X})$ in MHM-GNN.

If the largest graph in $\mathcal{D}_{\text{true}} \cup \mathcal{D}_{\text{noise}}$ has n nodes, directly computing the gradient of the loss $\nabla\mathcal{L}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ would take $\mathcal{O}(M|\mathcal{D}_{\text{true}}|^2 n^k)$ operations. Traditional Stochastic Gradient Descent (SGD) methods get rid of the dataset size $M|\mathcal{D}_{\text{true}}|^2$ term by uniformly sampling graph examples. Thus, naively optimizing the NCE loss with SGD would still require $\mathcal{O}(n^k)$ operations to compute $\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$. In what follows we rely on a stochastic optimization procedure that requires a finite-sample unbiased estimator of $\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$, where we can also control the estimator's variance with a hyperparameter. We show that the resulting stochastic optimization is theoretically sound by proving that it optimizes an upper bound of the original loss.

Estimating the MHM-GNN energy $\Phi(\mathbf{A}, \boldsymbol{X};\mathbf{W})$. To estimate $\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$, we need to first observe that —due to sparsity in real-world graphs— an arbitrary set of k nodes from a graph will induce an empty subgraph with high probability [43]. Therefore, to estimate $\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ with low variance, we focus on estimating it on *connected induced subgraphs* (CISes) [61], while assuming some constant high energy for disconnected subgraphs. To this end, if $\mathcal{C}^{(k)}_{\text{conn}}$ is the set of all k-node sets that induce a connected subgraph in G, we are now making the reasonable assumption

$$\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W}) = \sum_{C \in \mathcal{C}^{(k)}_{\text{conn}}} \phi(\mathbf{A}^{(C)}, \boldsymbol{X}^{(C)}; \mathbf{W}) + \text{constant}, \qquad (3)$$

where w.l.o.g. we assume the constant to be zero. Since enumerating all CISes is computationally intractable for arbitrary k [11], we introduce next a finite-sample unbiased estimator for $\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ of Equation (3) over CISes, denoted by $\widehat{\Phi}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$.

We start by presenting the concept of the higher-order network (k-HON) of a graph G and its variant called *collapsed node* HON (k-CNHON). An ordinary k-HON $G^{(k)}$ is a network where the nodes $V^{(k)}$ correspond to k-node CISes from G and edges $E^{(k)}$ connect two CISes that share $k-1$ nodes. On the other hand, a k-CNHON or $G^{(k,\mathcal{I})}$ is a multigraph where a subset of the nodes of $G^{(k)}$, $\mathcal{I} \subset V^{(k)}$, are collapsed into a single node in $G^{(k,\mathcal{I})}$. The collapsed node, henceforth denoted *the supernode*, is now node $v^{(k)}_{\mathcal{I}}$ in $G^{(k,\mathcal{I})}$. The edges in $G^{(k)}$ of the collapsed nodes $v \in \mathcal{I}$ among themselves, i.e., the edges in $\mathcal{I} \times \mathcal{I}$, do not exist in $G^{(k,\mathcal{I})}$. The edges between the collapsed nodes $v \in \mathcal{I}$ and other nodes $V \setminus \mathcal{I}$ are added to $G^{(k,\mathcal{I})}$ by replacing the endpoint v with endpoint $v^{(k)}_{\mathcal{I}}$, making $G^{(k,\mathcal{I})}$ a multigraph (a graph with multiple edges between the same two nodes). All the remaining edges in $G^{(k)}$ are preserved in $G^{(k,\mathcal{I})}$. In Figure 2 we show a graph and its k-CNHON with a Random Walk Tour (Definition 4) example. A formal definition is given in supplement.

Definition 4 (Random Walk Tour (RWT))**.** *Consider a simple random walk over a multigraph starting at node v_{init}. A Random Walk Tour (RWT) is represented by a sequence of nodes $\mathcal{T} = \{v_1, ..., v_t, v_{t+1}\}$ visited by the random walk such that $v_1 = v_{\mathit{init}}$, $v_{t+1} = v_{\mathit{init}}$ and $v_i \neq v_{\mathit{init}}$ $\forall\, 1 < i < t+1$.*

In this work, we construct the estimator $\widehat{\Phi}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ via *random walk tours* (RWTs) on the k-CNHON $G^{(k,\mathcal{I})}$ starting at the collapsed node $v^{(k)}_{\mathcal{I}}$ (i.e, $v_{\text{init}} = v^{(k)}_{\mathcal{I}}$ in Definition 4). As previously introduced and discussed in Avrachenkov et al. [2] and Teixeira et al. [61], increasing the number of tours and the supernode size allow for variance reduction. Using these insights, we propose the estimator $\widehat{\Phi}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$, whose properties are defined in Theorem 1.

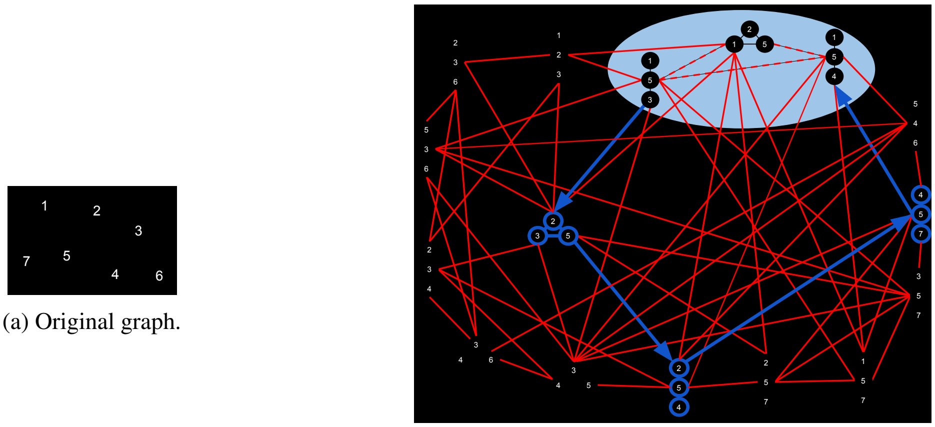
Theorem 1. *Let $G^{(k)}$ be the k-HON of a graph G, a set $\mathcal{I}$ of k-node sets that induce CISes in G (as described above) and $N^{(k)}(C)$ the set of neighbors of the corresponding node of CIS C in $G^{(k)}$. In addition, consider the sample-path $\mathcal{T}^r = (v^r_1, ..., v^r_{t^r}, v^r_{t^r+1})$ visited by the r-th RWT on $G^{(k,\mathcal{I})}$ starting from supernode $v^{(k)}_{\mathcal{I}}$, where v^r_i is the node reached at step i for $1 \leq r \leq q$ (Definition 4), and $q \geq 1$ is the number of RWTs. Since $\mathcal{T}^r$ is a RWT, $v^r_1 = v^{(k)}_{\mathcal{I}}$, $v^r_{t^r+1} = v^{(k)}_{\mathcal{I}}$ and $v^r_i \neq v^{(k)}_{\mathcal{I}} : 1 < i < t^r + 1$. The nodes $(v^r_2, ..., v^r_{t^r})$ in the sample path $\mathcal{T}^r$ have a corresponding sequence of induced k-node subgraphs in the graph G, denoted $\mathcal{T}^r_C = (C^r_i)^{t^r}_{i=2}$. Then, the estimator*

$$\widehat{\Phi}(\mathbf{A}, \boldsymbol{X}; \mathbf{W}) = \underbrace{\sum_{v \in \mathcal{I}} \phi(\mathbf{A}^{(v)}, \boldsymbol{X}^{(v)}; \mathbf{W})}_{\textit{Energy of } k\textit{-node CISes in } \mathcal{I} \textit{ (supernode)}} + \underbrace{\left(\frac{\sum_{u \in \mathcal{I}} |N^{(k)}(u) \setminus \mathcal{I}|}{q}\right) \sum_{r=1}^{q} \sum_{i=2}^{t^r} \frac{\phi(\mathbf{A}^{(C^r_i)}, \boldsymbol{X}^{(C^r_i)}; \mathbf{W})}{|N^{(k)}(C^r_i)|}}_{\textit{RWT-estimated energy of remaining } k\textit{-node CISes in } G}$$

is an unbiased and consistent estimator of $\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ in Equation (3) with constant=0.

The proof of Theorem 1 is in the supplement.

We can now replace $\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ in $\mathcal{L}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ with its estimator $\widehat{\Phi}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$, resulting in a loss estimate $\widehat{\mathcal{L}}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$. It follows from Theorem 1 and Jensen's inequality that our loss estimate is in expectation an upper bound to the true NCE loss, *i.e.* $\mathbb{E}_{\widehat{\Phi}}[\widehat{\mathcal{L}}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})] \geq \mathcal{L}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$. Moreover, note that using an estimator of this nature in higher-order GNNs, such as k-GNNs [39], does not allow for a bound in the loss estimation (please, see the supplement for further discussion). Note that the variance of $\widehat{\Phi}$ is controlled by the hyperparameter q, the number of tours.



(a) Original graph.

(b) k-CNHON with a supernode of size 3 highlighted in blue and an RWT example. Dashed red edges exist in the k-HON but are removed in the k-CNHON.

Figure 2: A graph and its corresponding k-CNHON with an RWT example.

5 Results

In this section, we evaluate the quality of the unsupervised motif representations learned by MHM-GNN over six datasets using two joint k-node transfer learning tasks. The tasks consider three citation networks, one coauthorship network and two product networks to show how the pre-trained motif representations consistently outperform pooling pre-trained node representations in predicting k-node hidden hyperedge labels in downstream tasks — details of these tasks are in the *Hyperedge Detection* and *DAG Leaf Counting* subsections.

A good k-node representation of a graph is able to capture hidden k-order relationships while only observing pairwise interactions. To this end, our tasks evaluate the quality of the unsupervised representations using two hidden hyperedge label prediction tasks. Using the pre-trained unsupervised learned representation as input, we train a simple logistic regression classifier to predict the hidden hyperedge label of a k-node set.

Datasets. We use the Cora, Citeseer and Pubmed [55] citation networks, the DBLP coauthorship network [69], the Steam [47] and the Rent the Runway [38] product networks (more details about the datasets are in the supplement). These datasets were chosen since they contain joint k-node information. In the coauthorship network, nodes correspond to authors and edges to the coauthorship of a paper, hidden from the training data we also have the papers and their corresponding author list. In the product networks, nodes correspond to products and an edge exists if the same user bought the two end-point products, hidden from the training data we have the list of products each user bought. In the citation network, nodes correspond to papers and edges to citations, hidden from the training data we also have the direction in which the citation occurred. These directions, paper author list and users purchase history which are hidden in the training data used by the unsupervised GNN and MHM-GNN representations, give us two transfer learning *k-node downstream tasks*, described in what follows.

Hyperedge Detection. This hyperedge task, inspired by Yadati et al. [69], creates a k-node hyperedge in a citation network whenever a paper cites $k-1$ other papers, in a coauthorship network whenever k authors write a paper together and in a product network whenever a user buys k products. Examples are in the citation networks k-size subgraphs with at least one node with degree $k-1$ and k-cliques in the other networks. Note how Yadati et al. [69] directly learns its representations from the hypergraph, a significantly easier task. The downstream classifier —a simple logistic regression classifier— uses the unsupervised pre-trained representations to classify whether a set of k nodes forms a (hidden) hyperege or not. This task allows us to compare the quality of the unsupervised node representations of GNNs against that of MHM-GNN.

DAG Leaf Counting. This task considers the citation networks. Again, baselines and MHM-GNN are trained over the undirected graphs. Due to the temporal order of citations, subgraphs correspond to Directed Acyclic Graphs (DAGs) in the directed structure. For a connected k-node induced subgraph in the directed graph, we want to predict the number of leaves of the resulting DAG. Again, the downstream classifier —a simple logistic regression classifier— uses the unsupervised pre-trained representations of a set of k-nodes to predict the exact number of leaves formed by the (hidden) k-node DAG. The number of leaves defines the number of influential papers in the k-node set.

MHM-GNN architecture. The energy function of MHM-GNN is as described in Equation (2), where we use a one-hidden layer feedforward network with LeakyReLU activations as ρ, a row-wise sum followed by also a one-hidden layer feedforward network with LeakyReLU activations as the READOUT function and a single layer GraphSAGE-mean Hamilton et al. [21] as the GNN.

Training the model. Since the datasets used in this section contain only one large graph for training —as in most of the real-world graph datasets— we need to construct a larger set of positive examples $\mathcal{D}_{\text{true}}$ to learn the distribution $\mathbb{P}(\mathbf{A}, \boldsymbol{X}|\mathbf{W})$. One way to overcome this issue is by subsampling the original large graph. While sampling smaller graphs that preserve the original graph properties, we can approximate the true $\mathbb{P}(\mathbf{A}, \boldsymbol{X}|\mathbf{W})$ distribution and control the complexity of $\widehat{\Phi}(\mathbf{A}, X; \mathbf{W}))$ (since tour return times are affected by the size of the graph). To this end, we construct $\mathcal{D}_{\text{true}}$ by subsampling the original graph with Forest Fire [35]. As for the noise distribution, we turn to the one used by Veličković et al. [64], where for each positive example we generate M negative samples by keeping the adjacency matrix and shuffling the feature matrix. This noise distribution allows us to keep structural properties of the graph, *e.g.* connectivity, while significantly changing how node features affect the distribution. We precisely describe all hyperparameters and hyperparameter tuning in the supplement.

Experimental setup. To evaluate the performance of the pre-trained MHM-GNN representations in the above downstream tasks, we first train the model accordingly for $k = 3$ and $k = 4$ motif sizes over all six datasets. In the citation and coauthorship networks, we have a single graph, thus these tasks require dividing the graph into training and test sets when evaluating the representations, such that the distribution of observed subgraphs is preserved. To this end, for each dataset, we perform min-cut clustering and use the two cuts for training and test data in the downstream task. For the product networks, to explore the inductive nature of our method, we create two graphs, one for training the models and one for testing the representations. For the Steam dataset, we train on the user-product data from 2014 and test considering the data from 2015. Similarly, for the Rent the Runway dataset, we train on data from 2016 and test on data from 2017. Tables 1 and 2 show our results for the Hyperedge Detection and Table 3 for the DAG Leaf Counting tasks. MHM-GNN uses motif sizes $k = 3, 4$. In the supplement, we also show results for $k = 5$. For each task (and k), we report the mean and the standard deviation of the balanced accuracy (mean recall of each class) achieved by logistic regression over five different runs. Furthermore, the pre-trained representations (baselines and our approach) have dimension 128. Additional implementation details and hyperparameter search can be found in the supplement.

Baselines. We evaluate the motif representations from MHM-GNN against two alternatives representing the k nodes using state-of-the-art unsupervised GNN representations: GraphSAGE [21] and Deep Graph Infomax (DGI) [64]. As a naive baseline, we compare against summing the original features from the nodes, *i.e.* a representation that ignores structural information. Moreover, we compare our pre-trained MHM-GNN representations with an untrained (random parameters) version of MHM-GNN. Further, since the citation and coauthorship networks consider single graphs, in the supplement we show results for these datasets with two prominent transductive node embedding methods [49, 19], evidencing how even in transductive settings node embeddings fail to capture joint k-node relationships.

Results. The hidden hyperedge downstream tasks are designed to better understand how well pre-trained unsupervised representations can capture joint k-node properties. A good joint k-node representation should be able to disentangle (hidden) polyadic relationships, even though they only have access to dyadic data. In our Hyperedge Detection task using pre-trained unsupervised representations, Tables 1 and 2 show that MHM-GNN representations consistently outperform GNN node representations across all datasets. In particular, MHM-GNN increases classification accuracy by up to 11% over the best-performing baseline. The results of our the DAG Leaf Counting task, shown in Table 3, reinforce that pre-trained unsupervised MHM-GNN representations can better capture joint k-node interactions. In particular, MHM-GNN representations observe classification accuracies by up to 24% in this downstream task.

Table 1: Balanced accuracy for the **Hyperedge Detection** task over subgraphs of size $k = 3$. We report mean and standard deviation over five runs.

Method	**Cora** $k = 3$	**Citeseer** $k = 3$	**Pubmed** $k = 3$	**DBLP** $k = 3$	**Steam** $k = 3$	**Rent the Runway** $k = 3$
GS-mean[21]	0.490 ± 0.03	0.509 ± 0.07	0.499 ± 0.00	0.560 ± 0.08	0.565 ± 0.01	0.665 ± 0.00
GS-max[21]	0.486 ± 0.04	0.493 ± 0.06	0.498 ± 0.00	0.569 ± 0.06	0.579 ± 0.02	0.667 ± 0.00
GS-lstm[21]	0.483 ± 0.04	0.486 ± 0.05	0.510 ± 0.02	0.585 ± 0.06	0.518 ± 0.01	0.518 ± 0.01
DGI[64]	0.487 ± 0.03	0.508 ± 0.07	0.509 ± 0.02	0.497 ± 0.00	0.588 ± 0.01	0.612 ± 0.00
Raw Features	0.499 ± 0.00	0.588 ± 0.00	0.502 ± 0.00	0.518 ± 0.00	0.534 ± 0.00	0.649 ± 0.00
MHM-GNN (Rnd)	0.498 ± 0.00	0.520 ± 0.05	0.498 ± 0.01	0.491 ± 0.01	0.571 ± 0.01	0.650 ± 0.00
MHM-GNN	**0.618** ± 0.03	**0.621** ± 0.01	**0.602** ± 0.06	**0.773** ± 0.02	**0.611** ± 0.01	**0.676** ± 0.00

Table 2: Balanced accuracy for the **Hyperedge Detection** task over subgraphs of size $k = 4$. We report mean and standard deviation over five runs.

Method	**Cora** $k = 4$	**Citeseer** $k = 4$	**Pubmed** $k = 4$	**DBLP** $k = 4$	**Steam** $k = 4$	**Rent the Runway** $k = 4$
GS-mean[21]	0.450 ± 0.11	0.544 ± 0.03	0.524 ± 0.05	0.530 ± 0.15	0.640 ± 0.03	0.851 ± 0.00
GS-max[21]	0.462 ± 0.09	0.538 ± 0.04	0.558 ± 0.05	0.511 ± 0.14	0.688 ± 0.01	0.855 ± 0.00
GS-lstm[21]	0.444 ± 0.09	0.536 ± 0.04	0.566 ± 0.06	0.653 ± 0.02	0.504 ± 0.01	0.546 ± 0.03
DGI[64]	0.463 ± 0.10	0.526 ± 0.04	0.549 ± 0.06	0.500 ± 0.00	0.664 ± 0.02	0.749 ± 0.03
Raw Features	0.529 ± 0.01	0.581 ± 0.00	0.498 ± 0.02	0.558 ± 0.01	0.535 ± 0.01	0.857 ± 0.00
MHM-GNN (Rnd)	0.490 ± 0.10	0.478 ± 0.03	0.510 ± 0.02	0.492 ± 0.02	0.679 ± 0.01	0.832 ± 0.01
MHM-GNN	**0.575** ± 0.03	**0.659** ± 0.08	**0.701** ± 0.10	**0.740** ± 0.05	**0.750** ± 0.00	**0.860** ± 0.00

Node representations and joint k-node graph tasks. Our experiments further validate the theoretical claims in Srinivasan and Ribeiro [57], that structural node representations are not capable of performing joint k-node tasks. That is, the inductive node representations baselines perform similarly to a random classifier in most settings in Tables 1 to 3. In contrast, the greater accuracy of MHM-GNN shows that joint k-node representations are informative.

Ablation study. As an ablation, we test whether our optimization in MHM-GNN improves the unsupervised joint k-node representations, when compared against random neural network weights. And while Tables 1 to 3 show that MHM-GNN with random weights perform well in the tasks, since they are effectively a type of motif feature, the higher accuracy of the optimized joint k-node representations shows that the optimized representations in MHM-GNN are indeed learned.

Table 3: Balanced accuracy for the **DAG Leaf Counting** task over subgraphs of size $k = 3$ and $k = 4$. We report mean and standard deviation over five runs.

Method	**Cora**		**Citeseer**		**Pubmed**	
	$k = 3$	$k = 4$	$k = 3$	$k = 4$	$k = 3$	$k = 4$
GS-mean[21]	0.468 ± 0.05	0.245 ± 0.06	0.492 ± 0.04	0.356 ± 0.02	0.502 ± 0.00	0.384 ± 0.03
GS-max[21]	0.467 ± 0.06	0.245 ± 0.07	0.486 ± 0.04	0.347 ± 0.01	0.499 ± 0.00	0.371 ± 0.03
GS-lstm[21]	0.473 ± 0.05	0.263 ± 0.07	0.482 ± 0.04	0.348 ± 0.01	0.507 ± 0.02	0.372 ± 0.03
DGI[64]	0.478 ± 0.05	0.278 ± 0.07	0.504 ± 0.06	0.350 ± 0.02	0.505 ± 0.02	0.362 ± 0.03
Raw Features	0.501 ± 0.01	0.325 ± 0.00	0.567 ± 0.00	0.380 ± 0.00	0.503 ± 0.00	0.339 ± 0.00
MHM-GNN (Rnd)	0.497 ± 0.00	0.327 ± 0.00	0.518 ± 0.04	0.319 ± 0.01	0.499 ± 0.01	0.343 ± 0.01
MHM-GNN	**0.593** ± 0.03	**0.452** ± 0.03	**0.606** ± 0.01	**0.469** ± 0.02	**0.626** ± 0.02	**0.475** ± 0.08

As opposed to node GNN representations and other non-compositional unsupervised graph representation approaches, MHM-GNN does not take graph-wide information as input. Thus, it is natural to wonder to what extent pre-trained MHM-GNN joint k-node representations are informative of the entire graph to which they belong to. Hence, in the supplement we consider whole-graph classification as the downstream task. In this setting, we show how composing (by pooling) MHM-GNN motif representations can perform better than non-compositional methods, further indicating how our learned motif representations can capture the underlying graph distribution $\mathbb{P}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$.

6 Conclusions

By combining hypergraph Markov networks, an unbiased finite-sample MCMC estimator, and graph representation learning, we introduced MHM-GNN, a new scalable class of energy-based representation learning methods capable of learning joint k-node representations over dyadic graphs in an *inductive unsupervised* manner. Finally, we show how pre-trained MHM-GNN representations achieve more accurate results in downstream joint k-node tasks. The energy-based optimization in this work allows for many extensions, such as designing different k-node subgraph representation learning methods, new subgraph function estimators for MHM-GNN's loss function, and formulating new joint k-node tasks.

Broader Impact

This work presents an unsupervised model together with a stochastic optimization procedure to generate k-node representations from graphs, such as online social networks, product networks, citation networks, coauthorship networks, etc. As is the case with any learning algorithm, it is susceptible to produce biased representations if trained with biased data. Moreover, although the representations might be bias free, the downstream task defined by the user might be biased and thus, also produce biased decisions.

Acknowledgments

This work was funded in part by the National Science Foundation (NSF) Awards CAREER IIS-1943364, CCF-1918483, and by the ARO, under the U.S. Army Research Laboratory contract number W911NF-09-2-0053, the Purdue Integrative Data Science Initiative, the Purdue Research Foundation, and the Wabash Heartland Innovation Network. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the sponsors. Further, we would like to thank Mayank Kakodkar for his invaluable feedback and discussion on subgraph function estimation.

References

[1] Aldous, D. and Fill, J. (1995). Reversible markov chains and random walks on graphs.

[2] Avrachenkov, K., Ribeiro, B., and Sreedharan, J. K. (2016). Inference in OSNs via Lightweight Partial Crawls. In *SIGMETRICS*, volume 44, pages 165–177, New York, New York, USA. ACM Press.

[3] Bai, S., Zhang, F., and Torr, P. H. (2019). Hypergraph convolution and hypergraph attention. *arXiv preprint arXiv:1901.08150*.

[4] Bakhtin, A., Deng, Y., Gross, S., Ott, M., Ranzato, M., and Szlam, A. (2020). Energy-based models for text. *arXiv preprint arXiv:2004.10188*.

[5] Barber, D. (2012). *Bayesian reasoning and machine learning*. Cambridge University Press.

[6] Battaglia, P. W., Hamrick, J. B., Bapst, V., Sanchez-Gonzalez, A., Zambaldi, V., Malinowski, M., Tacchetti, A., Raposo, D., Santoro, A., Faulkner, R., et al. (2018). Relational inductive biases, deep learning, and graph networks. *arXiv preprint arXiv:1806.01261*.

[7] Benson, A. R., Abebe, R., Schaub, M. T., Jadbabaie, A., and Kleinberg, J. (2018). Simplicial closure and higher-order link prediction. *Proceedings of the National Academy of Sciences*, 115(48):E11221–E11230.

[8] Bojchevski, A. and Günnemann, S. (2018). Deep Gaussian embedding of graphs: Unsupervised inductive learning via ranking. In *International Conference on Learning Representations*.

[9] Borgwardt, K. M., Ong, C. S., Schönauer, S., Vishwanathan, S., Smola, A. J., and Kriegel, H.-P. (2005). Protein function prediction via graph kernels. *Bioinformatics*, 21(suppl_1):i47–i56.

[10] Brémaud, P. (2013). *Markov chains: Gibbs fields, Monte Carlo simulation, and queues*, volume 31. Springer Science & Business Media.

[11] Bressan, M., Chierichetti, F., Kumar, R., Leucci, S., and Panconesi, A. (2017). Counting graphlets: Space vs time. In *Proceedings of the Tenth ACM International Conference on Web Search and Data Mining*, pages 557–566.

[12] Cai, J.-Y., Fürer, M., and Immerman, N. (1992). An optimal lower bound on the number of variables for graph identification. *Combinatorica*, 12(4):389–410.

[13] Christopher Morris, Gaurav Rattan, P. M. (2020). Weisfeiler and leman go sparse: Towards scalable higher-order graph embeddings. In *Graph Representation Learning and Beyond (GRL+, ICML 2020)*.

[14] Duvenaud, D. K., Maclaurin, D., Iparraguirre, J., Bombarell, R., Hirzel, T., Aspuru-Guzik, A., and Adams, R. P. (2015). Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pages 2224–2232.

[15] Erdös, P. and Rényi, A. (1959). On random graphs. *Publicationes Mathematicae Debrecen*, 6:290.

[16] Feng, Y., You, H., Zhang, Z., Ji, R., and Gao, Y. (2019). Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 3558–3565.

[17] Fey, M. and Lenssen, J. E. (2019). Fast graph representation learning with PyTorch Geometric. In *ICLR Workshop on Representation Learning on Graphs and Manifolds*.

[18] Frank, O. and Strauss, D. (1986). Markov graphs. *Journal of the american Statistical association*, 81(395):832–842.

[19] Grover, A. and Leskovec, J. (2016). node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 855–864.

[20] Gutmann, M. and Hyvärinen, A. (2010). Noise-contrastive estimation: A new estimation principle for unnormalized statistical models. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 297–304.

[21] Hamilton, W. L., Ying, R., and Leskovec, J. (2017a). Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*.

[22] Hamilton, W. L., Ying, R., and Leskovec, J. (2017b). Representation learning on graphs: Methods and applications. *IEEE Data Eng. Bull.*, 40(3):52–74.

[23] Hinton, G. E. (2002). Training products of experts by minimizing contrastive divergence. *Neural computation*, 14(8):1771–1800.

[24] Höfling, H. and Tibshirani, R. (2009). Estimation of sparse binary pairwise markov networks using pseudo-likelihoods. *Journal of Machine Learning Research*, 10(Apr):883–906.

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

[26] Kersting, K., Kriege, N. M., Morris, C., Mutzel, P., and Neumann, M. (2016). Benchmark data sets for graph kernels.

[27] Kindermann, R. and Snell, J. (1982). Markov Random Fields and Their Application, Providence, RI: Amer. *Math. Soc*.

[28] Kingma, D. P. and Ba, J. (2015). Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

[29] Kipf, T. N. and Welling, M. (2016). Variational graph auto-encoders. *NIPS Workshop on Bayesian Deep Learning*.

[30] Kipf, T. N. and Welling, M. (2017). Semi-supervised classification with graph convolutional networks. In *International Conference on Learning Representations (ICLR)*.

[31] Kohli, P., Torr, P. H., et al. (2009). Robust higher order potentials for enforcing label consistency. *International Journal of Computer Vision*, 82(3):302–324.

[32] Kolaczyk, E. D. and Csárdi, G. (2014). *Statistical analysis of network data with R*, volume 65. Springer.

[33] Lauritzen, S., Rinaldo, A., and Sadeghi, K. (2018). Random networks, graphical models and exchangeability. *Journal of the Royal Statistical Society: Series B (Statistical Methodology)*, 80(3):481–508.

[34] Lee, J. B., Rossi, R. A., Kong, X., Kim, S., Koh, E., and Rao, A. (2019). Graph convolutional networks with motif-based attention. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*, pages 499–508.

[35] Leskovec, J. and Faloutsos, C. (2006). Sampling from large graphs. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 631–636.

[36] Maron, H., Ben-Hamu, H., Shamir, N., and Lipman, Y. (2019). Invariant and equivariant graph networks. In *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*. OpenReview.net.

[37] Meng, C., Mouli, C. S., Ribeiro, B., and Neville, J. (2018). Subgraph pattern neural networks for high-order graph evolution prediction. In *AAAI*.

[38] Misra, R., Wan, M., and McAuley, J. (2018). Decomposing fit semantics for product size recommendation in metric spaces. In *Proceedings of the 12th ACM Conference on Recommender Systems*, pages 422–426.

[39] Morris, C., Ritzert, M., Fey, M., Hamilton, W. L., Lenssen, J. E., Rattan, G., and Grohe, M. (2019). Weisfeiler and Leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pages 4602–4609.

[40] Murphy, R. L., Srinivasan, B., Rao, V., and Ribeiro, B. (2019). Relational pooling for graph representations. In *ICML*. PMLR.

[41] Nair, V. and Hinton, G. E. (2009). Implicit mixtures of restricted boltzmann machines. In Koller, D., Schuurmans, D., Bengio, Y., and Bottou, L., editors, *Advances in Neural Information Processing Systems 21*, pages 1145–1152. Curran Associates, Inc.

[42] Narayanan, A., Chandramohan, M., Venkatesan, R., Chen, L., Liu, Y., and Jaiswal, S. (2017). graph2vec: Learning distributed representations of graphs. *arXiv preprint arXiv:1707.05005*.

[43] Newman, M. (2018). *Networks*. Oxford University Press.

[44] Orbanz, P. and Roy, D. M. (2014). Bayesian models of graphs, arrays and other exchangeable random structures. *IEEE transactions on pattern analysis and machine intelligence*, 37(2):437–461.

[45] Pan, S., Hu, R., Long, G., Jiang, J., Yao, L., and Zhang, C. (2018). Adversarially regularized graph autoencoder for graph embedding. In *IJCAI*, pages 2609–2615.

[46] Paszke, A., Gross, S., Massa, F., Lerer, A., Bradbury, J., Chanan, G., Killeen, T., Lin, Z., Gimelshein, N., Antiga, L., et al. (2019). Pytorch: An imperative style, high-performance deep learning library. In *Advances in neural information processing systems*, pages 8026–8037.

[47] Pathak, A., Gupta, K., and McAuley, J. (2017). Generating and personalizing bundle recommendations on steam. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1073–1076.

[48] Patil, P., Sharma, G., and Murty, M. N. (2020). Negative sampling for hyperlink prediction in networks. In *Pacific-Asia Conference on Knowledge Discovery and Data Mining*, pages 607–619. Springer.

[49] Perozzi, B., Al-Rfou, R., and Skiena, S. (2014). Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 701–710.

[50] Ranzato, M., Poultney, C., Chopra, S., and Cun, Y. L. (2007). Efficient learning of sparse representations with an energy-based model. In *Advances in neural information processing systems*, pages 1137–1144.

[51] Rossi, R. A., Ahmed, N. K., Koh, E., Kim, S., Rao, A., and Abbasi-Yadkori, Y. (2020). A structural graph representation learning framework.

[52] Rossi, R. A., Zhou, R., and Ahmed, N. K. (2018). Deep inductive network representation learning. In *Companion Proceedings of the The Web Conference 2018*, pages 953–960.

[53] Rowland, M. and Weller, A. (2017). Uprooting and rerooting higher-order graphical models. In *Advances in Neural Information Processing Systems*, pages 209–218.

[54] Samanta, B., De, A., Ganguly, N., and Gomez-Rodriguez, M. (2018). Designing random graph models using variational autoencoders with applications to chemical design. *arXiv preprint arXiv:1802.05283*.

[55] Sen, P., Namata, G., Bilgic, M., Getoor, L., Galligher, B., and Eliassi-Rad, T. (2008). Collective classification in network data. *AI magazine*, 29(3):93–93.

[56] Shi, G., Hönig, W., Yue, Y., and Chung, S.-J. (2020). Neural-swarm: Decentralized close-proximity multirotor control using learned interactions. *arXiv preprint arXiv:2003.02992*.

[57] Srinivasan, B. and Ribeiro, B. (2020). On the equivalence between positional node embeddings and structural graph representations. In *ICLR*.

[58] Sun, F.-Y., Hoffman, J., Verma, V., and Tang, J. (2020). Infograph: Unsupervised and semi-supervised graph-level representation learning via mutual information maximization. In *International Conference on Learning Representations*.

[59] Taylor, A., Singletary, A., Yue, Y., and Ames, A. (2019). Learning for safety-critical control with control barrier functions. *arXiv preprint arXiv:1912.10099*.

[60] Teh, Y. W., Welling, M., Osindero, S., and Hinton, G. E. (2003). Energy-based models for sparse overcomplete representations. *Journal of Machine Learning Research*, 4(Dec):1235–1260.

[61] Teixeira, C. H., Cotta, L., Ribeiro, B., and Meira, W. (2018). Graph pattern mining and learning through user-defined relations. In *2018 IEEE International Conference on Data Mining (ICDM)*, pages 1266–1271. IEEE.

[62] Tsitsulin, A., Mottin, D., Karras, P., Bronstein, A., and Müller, E. (2018). Netlsd: hearing the shape of a graph. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2347–2356.

[63] Veličković, P., Cucurull, G., Casanova, A., Romero, A., Lio, P., and Bengio, Y. (2017). Graph Attention Networks. *arXiv preprint arXiv:1710.10903*.

[64] Veličković, P., Fedus, W., Hamilton, W. L., Liò, P., Bengio, Y., and Hjelm, R. D. (2019). Deep Graph Infomax. In *International Conference on Learning Representations*.

[65] Wang, P., Lui, J., Ribeiro, B., Towsley, D., Zhao, J., and Guan, X. (2014). Efficiently estimating motif statistics of large networks. *ACM Transactions on Knowledge Discovery from Data (TKDD)*, 9(2):8.

[66] Weisfeiler, B. and Lehman, A. A. (1968). A reduction of a graph to a canonical form and an algebra arising during this reduction. *Nauchno-Technicheskaya Informatsia*, 2(9):12–16.

[67] Xu, K., Hu, W., Leskovec, J., and Jegelka, S. (2019). How powerful are graph neural networks? In *International Conference on Learning Representations*.

[68] Xu, Y., Rockmore, D., and Kleinbaum, A. M. (2013). Hyperlink prediction in hypernetworks using latent social features. In *International Conference on Discovery Science*, pages 324–339. Springer.

[69] Yadati, N., Nimishakavi, M., Yadav, P., Nitin, V., Louis, A., and Talukdar, P. (2019). Hypergcn: A new method for training graph convolutional networks on hypergraphs. In *Advances in Neural Information Processing Systems*, pages 1509–1520.

[70] Yanardag, P. and Vishwanathan, S. (2015). Deep graph kernels. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pages 1365–1374.

[71] Yoon, S.-e., Song, H., Shin, K., and Yi, Y. (2020). How much and when do we need higher-order informationin hypergraphs? a case study on hyperedge prediction. In *Proceedings of The Web Conference 2020*, pages 2627–2633.

[72] Zhang, M., Cui, Z., Jiang, S., and Chen, Y. (2018). Beyond link prediction: Predicting hyperlinks in adjacency space. In *AAAI*.

[73] Zhang, M., Cui, Z., Oyetunde, T., Tang, Y., and Chen, Y. (2016). Recovering metabolic networks using a novel hyperlink prediction method. *arXiv preprint arXiv:1610.06941*.

[74] Zheleva, E., Getoor, L., and Sarawagi, S. (2010). Higher-order graphical models for classification in social and affiliation networks. In *NIPS Workshop on Networks Across Disciplines: Theory and Applications*, volume 2.

A The Estimator $\widehat{\Phi}(\mathbf{A}, \boldsymbol{X};\mathbf{W})$

A.1 The k-CNHON network

Definition 5 (k-CNHON of G given $\mathcal{I}$, or $G^{(k,\mathcal{I})}$)**.** *Let $G^{(k)} = (V^{(k)}, E^{(k)})$ be the higher-order network (k-HON) of the input graph G, where each node $v^{(k)} \in V^{(k)}$ corresponds to a k-node set $C \in \mathcal{C}^{(k)}_{\mathit{conn}}$. For ease of understanding, we will levarege this correspondence and refer to nodes from $V^{(k)}$ with k-node sets from $\mathcal{C}^{(k)}_{\mathit{conn}}$ interchangeably. The edge set $E^{(k)}$ is defined such that $E^{(k)} = \{(v_i^{(k)}, v_j^{(k)}) : v_i^{(k)}, v_j^{(k)} \in \mathcal{C}^{(k)}_{\mathit{conn}}$ and $|v_i^{(k)} \cap v_j^{(k)}| = k-1\}$. Moreover, let $\mathcal{I}$ be a set of k-nodes sets $\mathcal{I} \subset \mathcal{C}^{(k)}_{\mathit{conn}}$. Then, a k-CNHON $G^{(k,\mathcal{I})} = (V^{(k,\mathcal{I})}, E^{(k,\mathcal{I})})$ with supernode $v_{\mathcal{I}}^{(k)}$ is a multigraph with node set $V^{(k,\mathcal{I})} = (V^{(k)} \backslash \mathcal{I}) \cup v_{\mathcal{I}}^{(k)}$ and edge multiset $E^{(k,\mathcal{I})} = E^{(k)} \backslash (E^{(k)} \cap (\mathcal{I} \times \mathcal{I})) \uplus \{(v_{\mathcal{I}}^{(k)}, v^{(k)}) : \exists\, (u^{(k)}, v^{(k)}) \in E^{(k)}, u^{(k)} \in \mathcal{I}$ and $v^{(k)} \notin \mathcal{I}\}$, where $\uplus$ is the multiset union operation.*

A.2 Proof of Theorem 1

To prove Theorem 1, we assume that $G^{(k,\mathcal{I})}$ has a stationary distribution π with

$$\pi(C_i) = \frac{|N^{(k)}(C_i)|}{\sum_{C' \in V^{(k)} \backslash \mathcal{I}} |N^{(k)}(C')| + \sum_{u \in \mathcal{I}} |N^{(k)}(u) \backslash \mathcal{I}|} \;\forall\, C_i \in V^{(k,\mathcal{I})} \backslash \{v_{\mathcal{I}}^{(k)}\},$$

and

$$\pi(v_{\mathcal{I}}^{(k)}) = \frac{\sum_{u \in \mathcal{I}} |N^{(k)}(u) \backslash \mathcal{I}|}{\sum_{C' \in V^{(k)} \backslash \mathcal{I}} |N^{(k)}(C')| + \sum_{u \in \mathcal{I}} |N^{(k)}(u) \backslash \mathcal{I}|}.$$

Fortunately, Wang et al. [65] showed that such a statement is true whenever $\mathcal{I}$ contains at least one k-node set from each connected component of G and if each such component contains at least one vertex which is not a part of any k-node set in $\mathcal{I}$ and is contained in more than 2 edges in G. First, we show that the estimate $\widehat{\Phi}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ of each tour is unbiased.

Lemma 1. *Let $\mathcal{T}_C^r = (C_i^r)_{i=2}^{t^r}$ be a k-node set chain formed by the samples from the r-th RWT on $G^{(k,\mathcal{I})}$ starting at the supernode $v_{\mathcal{I}}^{(k)}$. Then, $\forall r \geq 1$,*

$$\mathbb{E}\Big[\sum_{v \in \mathcal{I}} \phi(\mathbf{A}^{(v)}, \boldsymbol{X}^{(v)}; \mathbf{W}) + \Big(\sum_{u \in \mathcal{I}} |N^{(k)}(u) \backslash \mathcal{I}|\Big) \sum_{i=2}^{t^r} \frac{\phi(\mathbf{A}^{(C_i^r)}, \boldsymbol{X}^{(C_i^r)}; \mathbf{W})}{|N^{(k)}(C_i^r)|}\Big] = \Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W}), \quad (4)$$

assuming $\Phi(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ with zero constant.

Proof of Lemma 1. Let's first rewrite Equation (4) as

$$\Big(\sum_{u\in\mathcal{I}} |N^{(k)}(u)\backslash\mathcal{I}|\Big)\mathbb{E}\Big[\sum_{i=2}^{t^r} \frac{\phi(\mathbf{A}^{(C_i^r)}, \boldsymbol{X}^{(C_i^r)};\mathbf{W})}{|N^{(k)}(C_i^r)|}\Big] = \Phi(\mathbf{A},\boldsymbol{X};\mathbf{W}) - \sum_{v\in\mathcal{I}} \phi(\mathbf{A}^{(v)},\boldsymbol{X}^{(v)};\mathbf{W}). \tag{5}$$

Since the RWT starts at node $v_{\mathcal{I}}^{(k)}$, we may rewrite the expected value in Equation (5) as

$$\mathbb{E}\left[\sum_{i=2}^{t^r} \frac{\phi(\mathbf{A}^{(C_i^r)}, \boldsymbol{X}^{(C_i^r)};\mathbf{W})}{|N^{(k)}(C_i^r)|}\right] = \sum_{C_i\in\mathcal{C}^{(k)}_{\text{conn}}\backslash\mathcal{I}} \mathbb{E}\left[\mathbb{T}(C_i)\frac{\phi(\mathbf{A}^{(C_i)}, \boldsymbol{X}^{(C_i)};\mathbf{W})}{|N^{(k)}(C_i)|}\right], \tag{6}$$

where $\mathbb{T}(C)$ represents the number of times the RWT reaches state C.

Consider a renewal reward process with inter-renewal time distributed as t^r, $r \geq 1$ and reward as $\mathbb{T}(C_i^r)$. Further, note that the chain is positive recurrent, thus $\mathbb{E}[t^r] < \infty$, $\mathbb{E}[\mathbb{T}(C_i^r)] < \infty$ and $\mathbb{T}(C_i^r) < \infty$. Then, from the renewal reward theorem and the ergodic theorem [10] we have

$$\pi(C_i^r) = \mathbb{E}[t^r]^{-1}\mathbb{E}[\mathbb{T}(C_i^r)].$$

Moreover, it follows from Kac's formula [1] that $\mathbb{E}[t^r] = \frac{1}{\pi(v_{\mathcal{I}}^{(k)})}$. Therefore, Equation (6) can be rewritten as

$$\mathbb{E}\left[\sum_{i=2}^{t^r} \frac{\phi(\mathbf{A}^{(C_i^r)}, \boldsymbol{X}^{(C_i^r)};\mathbf{W})}{|N^{(k)}(C_i^r)|}\right] = \sum_{C_i\in\mathcal{C}^{(k)}_{\text{conn}}\backslash\mathcal{I}} \frac{\pi(C_i)\phi(\mathbf{A}^{(C_i)}, \boldsymbol{X}^{(C_i)};\mathbf{W})}{\pi(v_{\mathcal{I}}^{(k)})|N^{(k)}(C_i)|}. \tag{7}$$

Now, knowing the stationary distribution of $G^{(k,\mathcal{I})}$, we may simplify Equation (7) to

$$\mathbb{E}\left[\sum_{i=2}^{t^r} \frac{\phi(\mathbf{A}^{(C_i^r)}, \boldsymbol{X}^{(C_i^r)};\mathbf{W})}{|N^{(k)}(C_i^r)|}\right] = \frac{1}{\sum_{u\in\mathcal{I}}|N^{(k)}(u)\backslash\mathcal{I}|} \sum_{C_i\in\mathcal{C}^{(k)}_{\text{conn}}\backslash\mathcal{I}} \phi(\mathbf{A}^{(C_i)}, \boldsymbol{X}^{(C_i)};\mathbf{W}), \tag{8}$$

and replace it in Equation (5), concluding our proof. □

Proof of Theorem 1. By Lemma 1, linearity of expectation and knowing that each RWT is independent from the other tours by the Strong Markov Property, Theorem 1 holds. □

B Discussion of MHM-GNN properties

Conditional independence. Although HMNs factorize distributions, the potentials themselves do not provide information on conditional and marginal distributions. Rather, we need to analyze how every pair of variables interacts through all potentials. For the sake of simplicity, consider the model described in Definition 3 for undirected simple graphs, *i.e.* $\boldsymbol{A}_{ij} = \boldsymbol{A}_{ji}\ \forall\ (i,j) \in V^2, \mathbf{A}_{ii} = 0\ \forall\ i \in V$. If we set $k = 2$, each hyperedge will contain exactly one edge variable and two node variables, which is equivalent to assuming all edges are independent given their nodes' representations. Thus, for $k = 2$ MHM-GNN can recover edge-based models where representations don't use graph-wide information. Furthermore, if we allow the node representation to take graph-wide information, we can recover the recent Graph Neural Networks approaches [21, 29, 8]. If we opt for $k = 3$, a hyperedge defined by nodes i, j, l will contain the set of edge variables $\{\mathbf{A}_{ij}, \mathbf{A}_{il}, \mathbf{A}_{jl}\}$ and node variables $\{\boldsymbol{X}_{i,\cdot}, \boldsymbol{X}_{j,\cdot}, \boldsymbol{X}_{l,\cdot}\}$. Thus, a hyperedge will encompass only edge variables that share one endpoint. In this case, an edge variable $\mathbf{A}_{ij}$ is independent from $\{\mathbf{A}_{lm} : l, m \in V, \{l, m\} \cap \{i, j\} = \emptyset\}$ others given $\{\mathbf{A}_{il} : l \in V\} \cup \{\mathbf{A}_{im} : m \in V\} \cup \{\boldsymbol{X}_{i,\cdot} : i \in V\}$. Thus, MHM-GNN with $k = 3$ can be cast as an instance of the Markov random graphs class proposed by Frank and Strauss [18]. With $k \geq 4$, for every pair of edge variables $\mathbf{A}_{ij}, \mathbf{A}_{lm}$ there exists at least one $C \in \mathcal{C}^{(k)}$ such that $i, j, l, m \subseteq C$. Thus, there exists at least one hyperedge covering every pair of edge variables in the model, resulting in a fully connected hypergraph Markov Network. Therefore, for $k \geq 4$ the model does not assume any conditional independence between edge variables which, since subgraphs share edge variables, is a vital feature for joint k-node representations of graphs.

Exchangeability. Although with infinite data and an arbitrary energy function $\phi(\cdot, \cdot; \mathbf{W})$ MHM-GNN would learn a jointly exchangeable [44] distribution, we would like to impose such condition on the model, defining a proper graph model. Equivalently, we would like to guarantee that any two isomorphic graphs have the same probability under MHM-GNN. By definition, the sets of subgraphs from two isomorphic graphs are equivalent under graph isomorphism. Thus, if the subgraph energy function $\phi(\cdot, \cdot; \mathbf{W})$ is jointly exchangeable, the set of subgraph energies from two isomorphic graphs are equivalent. Since the sum operation is permutation invariant and the partition function is a constant, a jointly exchangeable subgraph energy function $\phi(\cdot, \cdot; \mathbf{W})$, such as a GNN, is enough to make MHM-GNN jointly exchangeable.

Exponential Random Graph Models (ERGMs). The form of MHM-GNN presented in Definition 3 resembles the general and classical expression of Exponential Random Graph Models (ERGMs) [32]. Indeed, as any energy-based network model, we can cast ours as an ERGM where the sufficient statistics are given by all k-size subgraphs. However, we do stress how any exchangeable graph model has a correspondent ERGM representation [33], even when it is not as clear as it in MHM-GNN.

C Additional Experiments and Implementation Details from Section 5

C.1 Results for $k = 5$

Here, we extend the results from Section 5 to a $k = 5$ setting in Table 5 and table 4. Due to the lack of papers with 5 authors (less than 10), we were not able to extend them to the DBLP dataset. Moreover, the conclusions from Section 5 also hold here. That is, MHM-GNN consistently outperforms the baselines. However, on Rent the Runway we see the raw features achieving the highest performance. That is, structural information does not seem to be relevant to this specific task. Nevertheless, we still see that MHM-GNN and GraphSAGE are the methods able to perform the task similarly to the raw features.

Table 4: Balanced accuracy for the **Hyperedge detection** task over subgraphs of size $k = 5$. We report mean and standard deviation over five runs.

Method	**Cora** $k = 5$	**Citeseer** $k = 5$	**Pubmed** $k = 5$	**Steam** $k = 5$	**Rent the Runway** $k = 5$
GS-mean[21]	0.447 ± 0.10	0.530 ± 0.03	0.697 ± 0.08	0.696 ± 0.07	0.933 ± 0.00
GS-max[21]	0.384 ± 0.09	0.543 ± 0.08	0.722 ± 0.06	0.765 ± 0.03	0.940 ± 0.00
GS-lstm[21]	0.422 ± 0.04	0.525 ± 0.03	0.736 ± 0.08	0.532 ± 0.05	0.557 ± 0.05
DGI[64]	0.504 ± 0.00	0.500 ± 0.00	0.500 ± 0.00	0.626 ± 0.11	0.827 ± 0.04
Raw Features	0.500 ± 0.00	0.513 ± 0.00	0.526 ± 0.00	0.602 ± 0.00	**0.944** ± 0.00
MHM-GNN (Rnd)	0.460 ± 0.05	0.453 ± 0.03	0.493 ± 0.07	0.748 ± 0.02	0.924 ± 0.00
MHM-GNN	**0.543** ± 0.06	**0.703** ± 0.04	**0.815** ± 0.10	**0.823** ± 0.00	0.943 ± 0.01

Table 5: Balanced accuracy for the **DAG Leaf Counting** task over subgraphs of size $k = 5$. We report mean and standard deviation over five runs.

Method	**Cora** $k = 5$	**Citeseer** $k = 5$	**Pubmed** $k = 5$
GS-mean[21]	0.223 ± 0.04	0.259 ± 0.02	0.284 ± 0.02
GS-max[21]	0.150 ± 0.07	0.263 ± 0.01	0.288 ± 0.02
GS-lstm[21]	0.214 ± 0.03	0.259 ± 0.00	0.295 ± 0.04
DGI[64]	0.236 ± 0.02	0.249 ± 0.00	0.249 ± 0.00
Raw Features	0.251 ± 0.05	0.266 ± 0.00	0.290 ± 0.00
MHM-GNN (Rnd)	0.231 ± 0.01	0.277 ± 0.02	0.244 ± 0.01
MHM-GNN	**0.363** ± 0.04	**0.364** ± 0.02	**0.330** ± 0.04

C.2 Hyperparameters and Hyperparameter Search for MHM-GNN

All MHM-GNN models were implemented in PyTorch [46] and PyTorch Geometric [17] with the Adam optimizer [28]. All hyperparameters were chosen to minimize training loss. For learning rate, we searched in {0.01, 0.001, 0.0001} finding the best learning rate to be 0.001 for all models. We used a single hidden layer feedforward network with LeakyReLU activations for both ρ and READOUT functions in all models. Furthermore, following GraphSAGE Hamilton et al. [21], for all models we do an L2 normalization in the motif representation layer, *i.e.* in the output of the READOUT function. Finally, for all models we use $M = 1$ negative example for each positive example. In what follows, we give specific hyperparameters and their search for experiments from Section 5, show results for transductive baselines, and introduce new whole-graph downstream tasks together with their specific hyperparameters and search as well.

C.3 Pre-trained HMH-GNN for k-node downstream tasks (Section 5)

MHM-GNN architecture. The energy function of MHM-GNN is as described in Equation (2), where we use a one-hidden layer feedforward network with LeakyReLU activations as ρ, a row-wise sum followed by also a one-hidden layer feedforward network with LeakyReLU activations as the READOUT function and a single layer GraphSAGE-mean Hamilton et al. [21] as the GNN, except for $k = 5$ in the citation networks where we used two layers of the GraphSAGE-mean GNN to achieve faster convergence in training.

Table 6: Results for transductive baselines in the **Hyperedge Detection** task over $k = 3$, $k = 4$ and $k = 5$ size subgraphs.

Method	**Cora**	**Citeseer**	**Pubmed**	**DBLP**
	$k = 3$	$k = 3$	$k = 3$	$k = 3$
node2vec[19]	0.534 ± 0.04	0.525 ± 0.02	0.501 ± 0.00	0.461 ± 0.05
node2vec[19] + Features	0.545 ± 0.01	0.534 ± 0.01	0.500 ± 0.00	0.479 ± 0.04
DeepWalk[49]	0.472 ± 0.02	0.433 ± 0.01	0.499 ± 0.00	0.481 ± 0.00
DeepWalk[49] + Features	0.512 ± 0.01	0.591 ± 0.01	0.502 ± 0.00	0.485 ± 0.02

(a) $(k = 3)$ Balanced accuracy for the **Hyperedge Detection** task over subgraphs of size $k = 3$. We report mean and standard deviation over five runs.

Method	**Cora**	**Citeseer**	**Pubmed**	**DBLP**
	$k = 4$	$k = 4$	$k = 4$	$k = 4$
node2vec[19]	0.537 ± 0.04	0.513 ± 0.03	0.504 ± 0.01	0.405 ± 0.01
node2vec[19] + Features	0.626 ± 0.03	0.540 ± 0.01	0.502 ± 0.00	0.548 ± 0.10
DeepWalk[49]	0.515 ± 0.07	0.494 ± 0.10	0.504 ± 0.01	0.460 ± 0.01
DeepWalk[49] + Features	0.597 ± 0.05	0.570 ± 0.01	0.516 ± 0.01	0.560 ± 0.03

(b) $(k = 4)$ Balanced accuracy for the **Hyperedge Detection** task over subgraphs of size $k = 4$. We report mean and standard deviation over five runs.

Method	**Cora**	**Citeseer**	**Pubmed**
	$k = 5$	$k = 5$	$k = 5$
node2vec[19]	0.446 ± 0.08	0.544 ± 0.08	0.623 ± 0.13
node2vec[19] + Features	0.519 ± 0.00	0.500 ± 0.00	0.502 ± 0.01
DeepWalk[49]	0.446 ± 0.07	0.568 ± 0.05	0.568 ± 0.13
DeepWalk[49] + Features	0.490 ± 0.01	0.523 ± 0.01	0.472 ± 0.11

(c) $(k = 5)$ Balanced accuracy for the **Hyperedge Detection** task over subgraphs of size $k = 5$. We report mean and standard deviation over five runs.

Subsampling positive examples. We use positive examples subsampled with Forest Fire [35] of size 100 for Cora, Citeseer and DBLP datasets, while for Pubmed, a larger network, we use examples of size 500. For Steam, a smaller network, we use 75 and for Rent the Runway, a mid-size network we use 150.

Number of tours. We did 80 tours for all datasets except Pubmed with $k = 4$, which due to a larger k-CNHON network, we did 120 tours. A small number of tours will result in high variance in the gradient which, as we observed, tends to impair the learning process. Therefore, we tested training models, each with a different fix number of tours, starting with 1 tour and increasing it 10 by 10 until we reached the reported number of tours, which results in training loss convergence.

Supernode size. To construct the supernode, we do a BFS on the k-HON of the original input graph, similarly to Teixeira et al. [61]. We have a parameter that controls the maximum number of subgraphs visited by the BFS, which we call supernode budget. This parameter was set to 100K for Pubmed with $k = 3$ and $k = 4$, 5K for Cora with $k = 3$ and $k = 4$, Citeseer with $k = 3$ and DBLP with $k = 3$, 10K for Citeseer with $k = 4$ and 50K for DBLP with $k = 4$. For Steam, we set to 1K for $k = 3$ and to 10K for $k = 4$. For Rent the Runway, we set to 10K for $k = 3$ and to 30K for $k = 4$. For $k = 5$, we used 50K in Cora, 75K in Citeseer, 120K in Pubmed, 50K in Steam and 100K in Rent the Runway. In the same way of tours, we started with a small supernode budget of 100 and increased it by 100 until we observed the tours being completed and the training loss converging.

Minibatch size. We used a minibatch size of 50 for Cora, Citeseer and Steam with $k = 3$ and 25 for Cora and Citesser with $k = 4$. For Pubmed, Rent the Runway and DBLP, larger networks, we used minibatches of size 40 for $k = 3$ and 10 for $k = 4$. For Steam, we used 20 for $k = 4$. Again, we tested small minibatch sizes, increasing them until we had training loss convergence and GPU memory space to use. For $k = 5$, we used a minibatch of size 5 in all datasets.

C.3.1 Transductive baselines

Since we defined the tasks from Section 5 over single graphs in the citation and couathorship networks, in Tables 6a to 6c, and Tables 7a to 7c we show for those datasets results for two prominent transductive node embedding methods, node2vec [19] and DeepWalk [49] together with concatenating the raw features to them, evidencing how even in transductive settings, transductive node embeddings fail to capture joint k-node relationships in most settings, performing similarly to the inductive approaches to node representations, thus, performing consistently worse than our MHM-GNN joint k-node representations.

Table 7: Results for transductive baselines in the **DAG Leaf Counting** task over $k = 3$, $k = 4$ and $k = 5$ size subgraphs.

Method	**Cora** $k = 3$	**Citeseer** $k = 3$	**Pubmed** $k = 3$
node2vec[19]	0.538 ± 0.05	0.546 ± 0.03	0.502 ± 0.01
node2vec[19] + Features	0.556 ± 0.02	0.527 ± 0.01	0.501 ± 0.00
DeepWalk[49]	0.466 ± 0.02	0.503 ± 0.06	0.499 ± 0.00
DeepWalk[49] + Features	0.543 ± 0.01	0.584 ± 0.00	0.503 ± 0.00

(a) ($k = 3$) Balanced accuracy for the **DAG Leaf Counting** task over subgraphs of size $k = 3$. We report mean and standard deviation over five runs.

Method	**Cora** $k = 4$	**Citeseer** $k = 4$	**Pubmed** $k = 4$
node2vec[19]	0.374 ± 0.06	0.329 ± 0.04	0.333 ± 0.00
node2vec[19] + Features	0.410 ± 0.04	0.388 ± 0.00	0.339 ± 0.00
DeepWalk[49]	0.322 ± 0.00	0.349 ± 0.04	0.339 ± 0.00
DeepWalk[49] + Features	0.349 ± 0.00	0.381 ± 0.00	0.345 ± 0.00

(b) ($k = 4$) Balanced Accuracy for the **DAG Leaf Counting** task over subgraphs of size $k = 4$. We report mean and standard deviation over five runs.

Method	**Cora** $k = 5$	**Citeseer** $k = 5$	**Pubmed** $k = 5$
node2vec[19]	0.265 ± 0.05	0.262 ± 0.03	0.298 ± 0.03
node2vec[19] + Features	0.263 ± 0.01	0.240 ± 0.02	0.259 ± 0.01
DeepWalk[49]	0.254 ± 0.02	0.240 ± 0.01	0.238 ± 0.05
DeepWalk[49] + Features	0.255 ± 0.00	0.269 ± 0.00	0.269 ± 0.01

(c) ($k = 5$) Balanced Accuracy for the **DAG Leaf Counting** task over subgraphs of size $k = 5$. We report mean and standard deviation over five runs.

C.4 Pre-trained MHM-GNN representations for whole-graph downstream tasks

In Section 5, we have seen that the motif representations learned by MHM-GNN can better predict hyperedge properties than existing unsupervised GNN representations. In the following experiments we investigate: Are MHM-GNN motif representations capturing graph-wide information (learning $\mathbb{P}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$)? To this end, inspired by Nair and Hinton [41]'s evaluation of RBM representations through supervised learning, we now investigate if MHM-GNN's pre-trained motif representations can do similarly or better than non-compositional methods that take graph-wide information in (inductive) whole-graph classification.

Datasets. We use four multiple graphs datasets, namely PROTEINS, ENZYMES, IMDB-BINARY and IMDB-MULTI [70, 26]. We are interested in evaluating whole-graph representations under two different scenarios, one where the nodes have high-dimensional feature vectors and the other where the nodes do not have features. To this end, we chose the two biological networks PROTEINS and ENZYMES, where nodes contain feature vectors of size 32 and 21 respectively and the social networks IMDB-BINARY and IMDB-MULTI where nodes do not have features. More details in Section D of this supplement.

Training the model. Since we have multiple graphs in our datasets, our set of positive graph examples is already given in the data, unlike in Section 5, where we had to subsample positives from a single graph. The negative examples still need to be sampled. For the biological networks, we used the same negative sampling approach used in Section 5. For the social networks, where the nodes do not have features, for each positive example, we uniformly at random add n edges to it, generating a negative sample (where n is the number of nodes in the graph).

Experimental setup. We equally divide the graphs in each dataset between training (unsupervised) and training+testing (supervised). We use two thirds of the graphs in the supervised dataset to train a logistic classifier for the downstream task over the graph's representation. We use a third of the supervised dataset to test the method's accuracy. The classification tasks used here are the same as in Borgwardt et al. [9] and Xu et al. [67]. Again, we set the representation dimension of both MHM-GNN and our baselines to 128. We show results for $k = 3, 4, 5$ motifs representations, $k = n$ whole-graph representations, and unsupervised GNN node representations. To create these representations, we tested both sum and mean pooling for MHM-GNN (except $k = n$) and all the node-based baselines. We report the best performance of each for a fair comparison.

Baselines. We compare MHM-GNN against *non-compositional methods*: pooling node representations from GraphSAGE and DGI, directly pooling node features, two recent whole-graph embedding methods, NetLSD [62] and graph2vec [42] and a recent unsuperved whole-graph representation, InfoGraph [58]. Apart from pooling node features, all methods input graph-wide information to their representations. Pooling node features is not applicable to the social networks, since they do not have such information. Additionally, DGI also generates a whole-graph representation to minimize the mutual entropy with the nodes' representations. Note how by setting

Table 8: Results for the whole-graph classification task evaluated over balanced accuracy. We report mean and standrad deviation over five runs.

Method	PROTEINS	ENZYMES	IMDB-BIN.	IMDB-MULT
GS-mean[21]	0.753 ± 0.01	0.435 ± 0.02	0.454 ± 0.01	0.347 ± 0.01
GS-max[21]	0.729 ± 0.01	0.400 ± 0.04	0.447 ± 0.01	0.360 ± 0.01
GS-lstm[21]	0.739 ± 0.01	0.404 ± 0.04	0.442 ± 0.00	0.342 ± 0.01
DGI (Nodes)[64]	0.743 ± 0.02	0.349 ± 0.04	0.469± 0.00	0.367 ± 0.02
DGI (Joint)[64]	0.756 ± 0.00	0.263 ± 0.03	0.568 ± 0.03	0.376 ± 0.01
Raw Features	0.665 ± 0.05	0.210 ± 0.02	–	–
NetLSD[62]	0.760 ± 0.00	0.250 ± 0.00	0.550 ± 0.00	0.430 ± 0.01
graph2vec[42]	0.685 ± 0.00	0.166 ± 0.00	0.507 ± 0.00	0.335 ± 0.00
InfoGraph[58]	0.690 ± 0.04	0.278 ± 0.04	**0.691** ± 0.04	**0.466** ± 0.02
MHM-GNN (Rnd) ($k = 3$)	0.733 ± 0.01	0.293 ± 0.02	0.586 ± 0.00	0.369 ± 0.001
MHM-GNN ($k = 3$)	**0.777** ± 0.01	**0.445** ± 0.01	0.586 ± 0.00	0.376 ± 0.00
MHM-GNN (Rnd) (k=4)	0.720 ± 0.02	0.229 ± 0.04	0.580 ± 0.00	0.371 ± 0.00
MHM-GNN ($k = 4$)	**0.780** ± 0.02	0.390 ± 0.04	0.621 ± 0.00	0.390 ± 0.002
MHM-GNN (Rnd) ($k = 5$)	0.722 ± 0.01	0.213 ± 0.03	0.580 ± 0.00	0.378 ± 0.005
MHM-GNN ($k = 5$)	**0.773** ± 0.01	0.326 ± 0.04	0.600 ± 0.01	0.397 ± 0.001
MHM-GNN (Rnd) ($k = n$)	0.704 ± 0.03	0.266 ± 0.02	**0.707** ± 0.02	**0.446** ± 0.005
MHM-GNN ($k = n$)	0.753 ± 0.00	0.327 ± 0.01	**0.694** ± 0.02	**0.451** ± 0.01

$k = n$, we consider the entire graph as a single motif and thus, learn a whole-graph representation. Again, all models were trained according to their original implementation.

Results. We show in Table 8 the results for whole-graph classification downstream tasks. For each task and each model, we report the mean and the standard deviation of the balanced accuracy (mean recall of each class) achieved by logistic regression over five different runs. We observe how our method consistently outperforms representations computed over the entire graph: the joint DGI approach, graph2vec and node representations pooling. Interestingly, we observe that when the graph has high-dimensional feature vectors of the nodes, pooling small motif representations better generalizes than all other methods to unseen graphs. On the other hand, we observe that using a joint whole-graph representation, either with $k = n$ in our model or with NetLSD or with InfoGRAPH, can perform better without node features. In fact, there is no significant difference between using a random and a trained model for the joint representation. It is known how a random GNN model simply assigns a unique representation to each class of graphs indistinguishable under the 1-WL test [67]. Therefore, for graphs without node features, assigning unique representations seems to be the best in this setting, which means that the tested graph embedding and unsupervised representation methods are not really capturing significant graph information. Overall, we observe that indeed motif representations are capable of representing the entire graph to which they belong and even give better results, evidencing how MHM-GNN is learning graph-wide information, *i.e.* capturing $\mathbb{P}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$ and how motif compositionality can explain networks functionality.

MHM-GNN architecture. We use the same ρ and READOUT functions as in Section 5, while changing the GNN to GIN Xu et al. [67] (which gave better validation results than the GAT, GCN, and GraphSAGE GNNs). Again, we use $M = 1$, *i.e.*, we sample one negative example for each positive sample. We show results of MHM-GNN for $k = 3, 4, 5, n$. For the estimator $\widehat{\Phi}(\mathbf{A}, \boldsymbol{X}; \mathbf{W})$, we perform 30 tours for every model and dataset.

GNN layer. We use a single-layer GIN Xu et al. [67] as the GNN layer in our method. For $k = n$, where the GNN is applied over large graphs, we used GIN with two layers. Note that we also tested GraphSAGE-mean, GCN and GAT GNN layers here, but GIN resulted in faster training loss convergence.

Number of tours. We did 30 tours for all datasets. Again, we tested training models, each with a different fix number of tours, starting with 1 tour and increasing 10 by 10 until we reached the reported number of tours, which results in training loss convergence.

Supernode size. We did a BFS with the maximum number of subgraphs visited as 5K for all models (and all k). Again, we started with a small supernode budget of 100 and increased it by 100 until we observed the tours being completed and the training loss converging.

Minibatch size. We used a minibatch size of 50 for ENZYMES and PROTEINS for all reported k. For IMDB-BINARY and IMDB-MULTI, which have larger networks we used a minibatch size of 10. Again, we tested small minibatch sizes and increased until we had training loss convergence and GPU memory to use.

Pooling functions. We tested both sum and mean pooling motif (our model) and node (baselines) representations for all models here. We observed that mean pooling performs the best for all models in all datasets, except for the ENZYMES dataset, where sum pooling performed the best for all models. Thus, Table 8 contain results with mean pooling for all models in the PROTEINS, IMDB-BINANRY and IMDB-MULTI datasets and sum pooling for all models in the ENZYMES dataset.

D Datasets

We present the datasets statistics in Table 9 and Table 10. For the PROTEINS and ENZYMES datasets, we added the node labels as part of the node features. For the DBLP, we subsampled (with Forest Fire) the original large network from Yadati et al. [69]. For the Steam graphs, we consider user-product relations from 2014 to create the training graph and data from 2015 to create the test graph. Similarly, we use 2016 data to create the Rent the Runway training graph and 2017 data to create the test graph. For both product networks, the node features we created are sparse bag-of-words from the user text reviews.

Table 9: Single graph datasets statistics.

Dataset	**Type**	**Nodes**	**Edges**	**Features**
Cora [55]	Citation Network	2,708	5,429	1,433
Citeseer [55]	Citation Network	3,327	4,732	3,703
Pubmed [55]	Citation Network	19,717	44,338	500
DBLP [69]	Coauthorship Network	4,309	12,863	1,425
Steam [47] (Train)	Product Network	1,098	7,839	775
Steam [47] (Test)	Product Network	1,322	7,547	775
Rent the Runway [38] (Train)	Product Network	2,985	55,979	1,475
Rent the Runway [38] (Test)	Product Network	5,003	67,365	1,475

Table 10: Multiple graphs datasets statistics.

Dataset	**Type**	**Graphs**	**Features**	**Classes**
PROTEINS [26]	Biological Network	1,113	32	2
ENZYMES [26]	Biological Network	600	21	6
IMDB-BINARY [26]	Social Network	1,000	0	2
IMDB-MULTI [26]	Social Network	1,500	0	3

E Related Work: Higher-order Graph Representations

In what follows, we review the existing approaches to higher-order graph representations in literature.

Higher-order graph representations. Morris *et. al* [39] showed how to expand the concept of a GNN, an approach based on the 1-WL algorithm [66], to a k-GNN, an approach based on the class of k-WL [12] algorithms, where instead of generating node representations, one can derive higher-order (k-size) representations later used to represent the entire graph. Although such approaches to represent entire graphs have been recently used in *supervised* graph classification tasks, how to systematically use them in an inductive unsupervised manner was not clear. Since edge-based models require factorizing over a 2-node representation, only 1-WL [30, 67, 21, 63] and 2-WL [39]-based GNNs can be used. Additionally, k-GNNs can be thought of as a GNN over an extended graph, where nodes are k-node tuples and edges exist between k-tuples that share exactly $k-1$ nodes. One could indeed think of applying an edge-based loss to the extended graph, where the nodes (k-node tuples) representations are given by a k-GNN. However, an edge-based model assumes independence among edges and an edge in the extended graph is repeated several times in the extended graphs, thus they are not independent. Finally, even if one could provide an unsupervised objective to k-GNNs, it would still require $\mathcal{O}(n^k(k\delta)L)$ steps to compute an L-layer k-GNN over a graph with n nodes and maximum degree δ. Due to the non-linearities in the READOUT function and in the neighborhood aggregations in k-GNNs, unbiased subgraph estimators such as the one presented in this work and neighborhood sampling technique such as the one from Hamilton et al. [21] would not provide an unbiased or a bounded loss estimation such as MHM-GNN does. Moreover, the more recent sparser version of k-GNNs [13] uses k-node tuple representations, instead of k-node subgraph represenations as in the original paper. Finally, MHM-GNN can take advantage of any graph representation method, including k-GNNs [39] and non-GNN approaches such the ones presented in Relational Pooling [40].

Sum-based subgraph representations. There has been recent work representing subgraphs by equating them with sets of node representations [22]. In general, these approaches use graph models able to generate node representations and then add a module on top to aggregate these individual representations in the downstream task. The most prominent efforts have treated subgraph representations as sums of the individual nodes' representations [22], namely sum-based techniques. These approaches do not rely on joint subgraph representations, *i.e.* subgraphs that share nodes will tend to have similar representations, constraining their representational power and thus relying more on the downstream task model.

Hypergraph models. In this work, we wish to learn a graph model through motif representations in the presence of standard dyadic (graph) data, *i.e.* we are only observing pairwise relationships. Therefore, we emphasize that hypergraph models, despite dealing with higher-order representations of graphs, require observing polyadic (hypergraph) data and therefore are not an alternative to the problem studied here.

Supervised learning with subgraphs. Meng et al. [37] made the first effort towards supervised learning with subgraphs, where the authors predict higher-order properties from temporal dyadic data, as opposed to the problem presented here, where we are are interested in *inductive unsupervised* learning of k-node sets from static graphs. Moreover, Meng *et. al* learned subgraph properties while optimizing a pseudo-likelihood function, *i.e.* ignoring the dependencies among different subgraphs in the loss function. Because different node sets share edge variables, it is vital to learn dependencies among them. Hence, here we presented the first graph model based on k-size motif structures trained with a proper Noise-Contrastive Estimation function, *i.e.* our model accounts for dependencies between every edge to represent k-size node sets.