ON THE EQUIVALENCE BETWEEN NODE EMBEDDINGS AND STRUCTURAL GRAPH REPRESENTATIONS

Anonymous authors
Paper under double-blind review

ABSTRACT

This work provides the first unifying theoretical framework for node embeddings and structural graph representations, bridging methods like matrix factorization and graph neural networks. Using invariant theory, we show that relationship between structural representations and node embeddings is analogous to that of a distribution and its samples. We prove that all tasks that can be performed by node embeddings can also be performed by structural representations and vice-versa. We also show that the concept of transductive and inductive learning is unrelated to node embeddings and graph representations, clearing another source of confusion in the literature. Finally, we introduce new practical guidelines to generating and using node embeddings, which further augments standard operating procedures used today.

1 INTRODUCTION

The theory of *structural* graph representations is a recently emerging field. It creates a link between relational learning and invariant theory. Interestingly, or rather unfortunately, there is no unified theory connecting node embeddings —low-rank matrix approximations, factor analysis, latent semantic analysis, etc.— with structural graph representations. Instead, conflicting interpretations have manifested over the last few years, that further confound practitioners and researchers alike.

For instance, consider the direction, *word embeddings* $\rightarrow$ *structural representations*, where the structural equivalence between *men* $\rightarrow$ *king* and *women* $\rightarrow$ *queen* is described as being obtained by just adding or subtracting their node embeddings (Arora et al., 2016; Mikolov et al., 2013). *Hence, can all node embeddings provide structural relationships akin to word analogies?* In the opposite direction, *structural representations* $\rightarrow$ *node embeddings*, graph neural networks (GNNs) are often optimized to predict edges even though their structural node representations are provably incapable of performing the task. For example, the node representations of the *lynx* and the *orca* in Figure 1 are indistinguishable due to an isomorphic equivalence between the nodes, making any edge prediction task that distinguishes the edges of *lynx* and *orca* a seemly futile exercise. *Hence, are structural representations in general —and GNNs in particular— fundamentally incapable of performing link (dyadic) and multi-ary (polyadic) predictions tasks?*

Confirmation bias has seemingly appeared to thwart recent efforts to bring node embeddings and structural representations in a single overarching framework. Preconceived notions of the two being fundamentally different have been reinforced in the existing literature, arguing they belong in different applications: Node embeddings would find applications in multi-ary relationships such as link prediction, clustering, and natural language processing and knowledge acquisition through word and entity embeddings. Structural representations would find applications in node classification, graph classification, and role discovery. A unified theory is required if we wish to eliminate these artificial boundaries and better cross-pollinate node embeddings and structural representations in novel techniques.

Contributions: In this work we use invariant theory to develop a unified theoretical framework that clarifies the differences between node embeddings and structural representations and emphasizes their correspondence. More specifically, (a) we show that structural representations and node embeddings have the same relationship as distributions and their samples; (b) we prove that all tasks that can be performed by node embeddings can also be performed by structural representations and vice-versa. Moreover, (c) we introduce new guidelines to creating and using node embeddings,

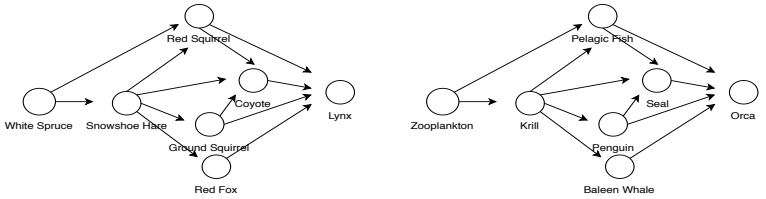


Figure 1: A food web example showing two disconnected components - the boreal forest (Stenseth et al., 1997) and the antarctic (Bates et al., 2015). The node embedding of the lynx and the orca can be different while their structural representation must be the same (due to the isomorphism).

which we hope will replace the less-than-optimal standard operating procedures used today. Finally, (d) we show that the concepts of *transductive* and *inductive* learning —commonly used to describe relational methods— are unrelated to node embeddigns and structural representations.

2 PRELIMINARIES

This section introduces some basic definitions, attempting to keep the mathematical jargon in check as much as we can, sometimes even sacrificing generality for clarity. We recommend Bloem-Reddy & Teh (2019) for a more formal description of some of the definitions in this section.

Definition 1 (Graph)**.** *We consider either a directed or an undirected attributed graph, denoted by $G = (V, E, \boldsymbol{X}, \mathbf{E})$, where V is a set of $n = |V|$ vertices, E is the set of edges in $V \times V$, with matrix $\boldsymbol{X} \in \mathbb{R}^{n \times k}$ and 3-mode tensor $\mathbf{E} \in \mathbb{R}^{n \times n \times k'}$ representing the node and edge features, respectively. The edge set has an associated adjacency matrix $\boldsymbol{A} \in \{0, 1\}^{n \times n}$. In order to simplify notation, we will compress $\mathbf{E}$ and $\boldsymbol{A}$ into a single tensor $\mathbf{A} \in \mathbb{R}^{n \times n \times (k'+1)}$. When explicit vertex and edge features and weights are unavailable, we will consider $\boldsymbol{X} = \mathbf{1}\mathbf{1}^T$ and $\mathbf{A} = \boldsymbol{A}$, where $\mathbf{1}$ is a $n \times 1$ vector of ones. We will abuse notation and denote the graph as $G = (\mathbf{A}, \boldsymbol{X})$. Without loss of generality, we define number the nodes in $V = \{1, \dots, n\}$ following the same ordering as the adjacency tensor $\mathbf{A}$ and the rows in $\boldsymbol{X}$. We denote $\vec{S}$ and a vector of the elements in $S \in \mathcal{P}^\star(V)$ sorted in ascending order, where $\mathcal{P}^\star(V)$ is the power set of V without the empty set.*

One of the most important operators in our mathematical toolkit will be that of a permutation action, orbits, $\mathcal{G}$-invariance, and $\mathcal{G}$-equivariance:

Definition 2 (Permutation action π)**.** *A permutation action π is a function that acts on any vector, matrix, or tensor defined over the nodes V, e.g., $(\boldsymbol{Z}_i)_{i \in V}$, and outputs an equivalent vector, matrix, or tensor with the order of the nodes permuted. We define Π_n as the set of all $n!$ such permutation actions.*

Definition 3 (Orbits)**.** *An orbit is the result of a group action Π_n acting on elements of a group correspond to bijective transformations of the space that preserve some structure of the space. The orbit of an element $\omega \in \Omega$ is the set of equivalent elements under action Π_n, i.e., $\Pi_n(x) = \{\pi(x) \mid \pi \in \Pi_n\}$.*

Definition 4 ($\mathcal{G}$-equivariant and $\mathcal{G}$-invariant functions)**.** *Let Σ be the set of all possible attributed graphs G. More formally, Σ is the set of all tuples $(\mathbf{A}, \boldsymbol{X})$ with all-size adjacency tensors $\mathbf{A}$ and corresponding node attributes $\boldsymbol{X}$. A function $g : \Sigma \to \mathbb{R}^{n \times \cdot}$ is $\mathcal{G}$-equivariant w.r.t. valid permutations of the nodes V, whenever any permutation action $\pi \in \Pi_n$ in the Σ symmetric space is associated with the same permutation action of the nodes in the $\mathbb{R}^{n \times \cdot}$ symmetric space. A function $g : \Sigma \to \mathbb{R}^{\cdot}$ is $\mathcal{G}$-invariant whenever it is invariant to any permutation action $\pi \in \Pi_n$ in the Σ symmetric space.*

Definition 5 (Graph orbits & graph isomorphism)**.** *Let $G = (\mathbf{A}, \boldsymbol{X})$, and let $\Pi_n(G) = \{(\mathbf{A}', \boldsymbol{X}') : (\mathbf{A}', \boldsymbol{X}') = (\pi(\mathbf{A}), \pi(\boldsymbol{X})), \forall \pi \in \Pi_n\}$ be the set of all equivalent (isomorphic) graphs under the permutation action π. Two graphs $G_1 = (\mathbf{A}_1, \boldsymbol{X}_1)$ and $G_2 = (\mathbf{A}_2, \boldsymbol{X}_2)$ are said isomorphic iff $\Pi_n(G_1) = \Pi_n(G_2)$.*

Definition 6 (Node orbits & node isomorphism)**.** *The equivalence classes of the vertices of a graph G under the action of automorphisms are called vertex orbits. If two nodes are in the same node orbit, we say that they are isomorphic. More precisely, two nodes $i, j \in V$ of an attributed graph $G = (\mathbf{A}, \boldsymbol{X})$ are in the same orbit iff in any permutation π of the nodes, switching positions of the nodes i and j into a new permutation $\pi'_{i \leftrightarrow j}$ will result in $\pi(\mathbf{A}) = \pi'_{i \leftrightarrow j}(\mathbf{A})$, $\pi(\boldsymbol{X}) = \pi'_{i \leftrightarrow j}(\boldsymbol{X})$.*

In Figure 1, the lynx and the orca are isomorphic (they have the same node orbits). We now generalize Definition 6 to subsets of nodes $S \in \mathcal{P}^\star(V)$, where $\mathcal{P}^\star(V)$ is the power set of V without the empty set.

Definition 7 (Vertex subset orbits and joint isomorphism)**.** *The equivalence classes of k-sized subsets of vertices $S \in \mathcal{P}^\star(V)$ of a graph G under the action of automorphisms between the subsets are called vertex subset orbits, $k \geq 2$. If two proper subsets $S_1, S_2 \in \mathcal{P}^\star(V) \backslash V$ are in the same vertex subset orbit, we say they are jointly isomorphic.*

Next we define the relationship between structural representations and node embeddings.

3 A Unifying Theoretical Framework of Node Embeddings and Structural Representations

How are node embeddings and structural representations related? This section starts with a familiar, albeit naïve, view of the differences between node embeddings and structural representations, preparing the groundwork to later broadening and rectifying these into precise model-free mathematical statements using invariant theory. This broadening is needed since model-free node embeddings need not be related to *node closeness* in the graph (or to lower dimensional projections for that matter), as it is impossible to have a model-free definition of *closeness*.

A familiar interpretation: Node embeddings are often seen as a lower-dimensional projection of the rows and columns of the adjacency matrix $\boldsymbol{A}$ from $\mathbb{R}^n$ to $\mathbb{R}^d$, $d < n$, that preserves relative positions of the nodes in a graph (Graham & Winkler, 1985; Linial et al., 1995); for instance, in Figure 1, the lynx and the coyote would have close node embeddings, while the node embeddings of the lynx and the orca would be significantly different. Node embeddings are often seen as encoding the fact that the lynx and the coyote are part of a tightly-knit community, while the lynx and orca belong to distinct communities. The *structural representation* of a node, on the other hand, shows which nodes have similar roles (structural similarities) on a graph; for instance, the lynx and the orca in Figure 1 must have the same structural representation, while the lynx and the coyote likely have different structural representations. The lynx, like the orcas, is a top predator in the food web while the coyote is not a top predator.

The frailty of the familiar interpretation: The interpretation of node embeddings above must be tied to a model that defines *closeness*. A general theory of node embeddings should be model-free. Unfortunately, one cannot define *closeness* without a model. Hence, in the remainder of this paper, we abandon this familiar interpretation in favor of a deeper mathematical definition, ultimately achieved in Definition 12.

Roadmap: In what follows, we restate some existing model-free definitions of *structural graph representation* and introduce some new ones. Then, we introduce a model-free definition of node embeddings. We will retain the terminology *node embedding* for historical reasons, even though our *node embedding* need not be an embedding (a projection into lower dimensional space).

3.1 On Structural Representations

In what follows we use the terms link and edge interchangeably. Proofs are left to the Appendix.

Definition 8 (Structural node representations)**.** *The structural representation of node $v \in V$ in a graph $G = (\mathbf{A}, \boldsymbol{X})$ is the $\mathcal{G}$-invariant representation $\Gamma(v, \mathbf{A}, \boldsymbol{X})$, where $\Gamma : V \times \Sigma \to \mathbb{R}^d$, $d \geq 1$, such that $\forall u \in V$, $\Gamma(u, \mathbf{A}, \boldsymbol{X}) = \Gamma(\pi(u), \pi(\mathbf{A}), \pi(\boldsymbol{X}))$ for all permutation actions $\forall \pi \in \Pi_n$. Moreover, for any two isomorphic nodes $u, v \in V$, $\Gamma(u, \mathbf{A}, \boldsymbol{X}) = \Gamma(v, \mathbf{A}, \boldsymbol{X})$.*

Definition 9 (Most-expressive structural node representations $\Gamma^\star$)**.** *A structural representation of a node $v \in V$, $\Gamma^\star(v, \mathbf{A}, \boldsymbol{X})$, is most-expressive iff, $\forall u \in V$, there exists a bijective measurable map between $\Gamma^\star(u, \mathbf{A}, \boldsymbol{X})$ and the orbit of node u in $G = (\mathbf{A}, \boldsymbol{X})$ (Definition 7).*

Trivially, by Definitions 6 and 9, two graphs $G_1 = (\mathbf{A}_1, \boldsymbol{X}_1)$ and $G_2 = (\mathbf{A}_2, \boldsymbol{X}_2)$ are isomorphic (Definition 5) iff the most-expressive structural node representations $(\Gamma^\star(u, \mathbf{A}_1, \boldsymbol{X}_1))_{u \in V}$ and $(\Gamma^\star(v, \mathbf{A}_2, \boldsymbol{X}_2))_{v \in V}$ are the same up to a valid permutation $\pi \in \Pi_n$ of the nodes.

We now describe the relationship between structural node representations and node isomorphism.

Lemma 1. *Two nodes $v, u \in V$, have the same most-expressive structural representations $\Gamma^\star(v, \mathbf{A}, \boldsymbol{X}) = \Gamma^\star(u, \mathbf{A}, \boldsymbol{X})$ iff u and v are isomorphic nodes in $G = (\mathbf{A}, \boldsymbol{X})$.*

Having described representation of nodes, we now generalize these representations to subsets of V.

Definition 10 (Joint structural representation Γ)**.** *A joint structural representation $\Gamma : \mathcal{P}^\star(V) \times \Sigma \to \mathbb{R}^d$, $d \geq 1$, where $\mathcal{P}^\star(V)$ is the power set of V excluding the empty set. Furthermore, Γ is $\mathcal{G}$-invariant over all node subsets, i.e., $\forall S \in \mathcal{P}^\star(V)$ and $\forall(\mathbf{A}, \boldsymbol{X}) \in \Sigma$, it must be that $\Gamma(\vec{S}, \mathbf{A}, \boldsymbol{X}) = \Gamma(\pi(\vec{S}), \pi(\mathbf{A}), \pi(\boldsymbol{X}))$ for all permutation actions $\forall \pi \in \Pi_n$. Moreover, for any two isomorphic subsets $S, S' \in \mathcal{P}^\star(V)$, $\Gamma(\vec{S}, \mathbf{A}, \boldsymbol{X}) = \Gamma(\vec{S'}, \mathbf{A}, \boldsymbol{X})$.*

We now mirror Definition 9 in our generalization of Γ:

Definition 11 (Most-expressive joint structural representations $\Gamma^\star$)**.** *A structural representation $\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})$ of a non-empty subset $S \in \mathcal{P}^\star(V)$, of a graph $(\mathbf{A}, \boldsymbol{X}) \in \Sigma$, is most-expressive iff, there exists a bijective measurable map between $\Gamma^\star(\vec{U}, \mathbf{A}, \boldsymbol{X})$ and the orbit of U in G (Definition 7), $\forall U \in \mathcal{P}^\star(V)$ and $\forall(\mathbf{A}, \boldsymbol{X}) \in \Sigma$.*

Note, however, the failure to represent the link (*lynx*, *coyote*) in Figure 1 using the most-expressive node representations of the *lynx* and the *coyote*. A link needs to be represented by a joint representation of two nodes. For instance, we can easily verify from Definition 11 that $\Gamma^\star((\textit{lynx,coyote}), \mathbf{A}, \boldsymbol{X}) \neq \Gamma^\star((\textit{orca,coyote}), \mathbf{A}, \boldsymbol{X})$, even though $\Gamma^\star(\textit{lynx}, \mathbf{A}, \boldsymbol{X}) = \Gamma^\star(\textit{orca}, \mathbf{A}, \boldsymbol{X})$.

Next we show that joint prediction tasks only require joint structural representations.

Theorem 1. *Let $\mathcal{S} \subseteq \mathcal{P}^\star(V)$ be a set of non-empty subsets of V. Let $\boldsymbol{Y}(\mathcal{S}, \mathbf{A}, \boldsymbol{X}) = (Y(\vec{S}, \mathbf{A}, \boldsymbol{X}))_{S \in \mathcal{S}}$ be a sequence of random variables defined over the sets $S \in \mathcal{S}$ of a graph $G = (\mathbf{A}, \boldsymbol{X})$, such that $Y(\vec{S_1}, \mathbf{A}, \boldsymbol{X}) \overset{d}{=} Y(\vec{S_2}, \mathbf{A}, \boldsymbol{X})$ for any two jointly isomorphic subsets $S_1, S_2 \in \mathcal{S}$ (Definition 7), where $\overset{d}{=}$ means equality in their marginal distributions. Then, there exists a measurable function φ such that, $\boldsymbol{Y}(\mathcal{S}, \mathbf{A}, \boldsymbol{X}) \overset{a.s.}{=} (\varphi(\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X}), \epsilon_S))_{S \in \mathcal{S}}$, where ϵ_S is a pure source of random noise from a joint distribution $p((\epsilon_{S'})_{\forall S' \in \mathcal{S}})$ independent of $\mathbf{A}$ and $\boldsymbol{X}$.*

Theorem 1 extends Theorem 12 of Bloem-Reddy & Teh (2019) to all subsets of nodes, $S \in \mathcal{P}^\star(V)$, also showing that any prediction task that can be defined over S, requires only a most-expressive joint structural representation over S. For instance, any task with $|S| = 2$ predicting a missing link (u, v) on a graph $G = (\mathbf{A}, \boldsymbol{X})$, requires only the most-expressive structural representation $\Gamma^\star((u, v), \mathbf{A}, \boldsymbol{X})$. Note that, in order to predict directed edges, we must use $\Gamma^\star((u, v), \mathbf{A}, \boldsymbol{X})$ to also predict the edge's direction: $\rightarrow$, $\leftarrow$, $\leftrightarrow$. Theorem 1 also includes node tasks for $|S| = 1$, hyperedge tasks for $2 < |S| < n$, and graph-wide tasks for $S = V$.

Remark 1 (GNNs and link prediction)**.** *Even though structural node representations of GNNs are not able to predict edges, GNNs are often still optimized to predict edges (e.g., (Hamilton et al., 2017a; Xu et al., 2018)) in transfer learning tasks. This optimization objective guarantees that any small topological differences between two nearly-isomorphic nodes without an edge will be amplified, while differences between nodes with an edge will be minimized. Hence, the topological differences in a close-knit community will be minimized in the representation. This procedure should work well for node classification tasks in homophilic networks (where classes tend to be clustered).*

We now turn our attention to node embeddings and their relationship with joint representations.

3.2 On Node Embeddings

Definition 12 (Node Embeddings)**.** *The node embeddings of a graph $G = (\mathbf{A}, \boldsymbol{X})$ are defined as joint samples of random variables $(\boldsymbol{Z}_i)_{i \in V} | \mathbf{A}, \boldsymbol{X} \sim p(\cdot | \mathbf{A}, \boldsymbol{X})$, $\boldsymbol{Z}_i \in \mathbb{R}^d$, $d \geq 1$, where $p(\cdot | \mathbf{A}, \boldsymbol{X})$ is a $\mathcal{G}$-equivariant probability distribution on $\mathbf{A}$ and $\boldsymbol{X}$, that is, $\pi(p(\cdot | \mathbf{A}, \boldsymbol{X})) = p(\cdot | \pi(\mathbf{A}), \pi(\boldsymbol{X}))$ for any permutation $\pi \in \Pi_n$.*

Essentially, Definition 12 says that the probability distribution $p(\boldsymbol{Z} | \mathbf{A}, \boldsymbol{X})$ of a node embedding $\boldsymbol{Z}$ must be $\mathcal{G}$-equivariant on $\mathbf{A}$ and $\boldsymbol{X}$. This is the only property we require to define a node embedding. Next, we show that the node embeddings given by Definition 12 cover a wide range of embedding methods in the literature.

Corollary 1. *The node embeddings in Definition 12 encompass embeddings given by matrix and tensor factorization methods —such as Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), implicit matrix factorization (a.k.a. word2vec)–, latent embeddings given by Bayesian graph models —such as Probabilistic Matrix Factorizations (PMFs) and variants—, variational autoencoder methods and graph neural networks that use random lighthouses to extract node embedddings.*

The proof of Corollary 1 is in the Appendix, along with the references of each of the methods mentioned in the corollary. The output of some of the methods described in Corollary 1 is deterministic, and for those, the probability density $p(\boldsymbol{Z}|\mathbf{A}, \boldsymbol{X})$ is a Dirac delta. In practice, however, even deterministic methods use algorithms whose outputs depend on randomized initial conditions, which will also satisfy Corollary 1.

We now show that permutation equivariance implies two isomorphic nodes (or two subsets of nodes) must have the same marginal distributions over $\boldsymbol{Z}$:

Lemma 2. *The permutation equivariance of p in Definition 12 implies that, if two proper subsets of nodes $S_1, S_2 \in \mathcal{P}^\star(V) \backslash V$ are isomorphic, then their marginal node embedding distributions must be the same up to a permutation, i.e., $p((\boldsymbol{Z}_i)_{i\in S_1}|\mathbf{A}, \boldsymbol{X}) = \pi(p((\boldsymbol{Z}_j)_{j\in S_2}|\mathbf{A}, \boldsymbol{X}))$ for some appropriate permutation $\pi \in \Pi_n$.*

Hence, the critical difference between the structural node representation vector $(\Gamma(v, \mathbf{A}, \boldsymbol{X}))_{v\in V}$ in Definition 8 and node embeddings $\boldsymbol{Z}$ in Definition 12, is that the vector $(\Gamma(v, \mathbf{A}, \boldsymbol{X}))_{v\in V}$ *must be* $\mathcal{G}$-equivariant while $\boldsymbol{Z}$ *need not be* —even though $\boldsymbol{Z}$'s distribution must be $\mathcal{G}$-equivariant. This seemly trivial difference has tremendous consequences, which we explore in the reminder of this section.

Next, we show that node embeddings $\boldsymbol{Z}$ cannot have any extra information about G that is not already contained in a most-expressive structural representation $\Gamma^\star$.

Theorem 2 (The statistical equivalence between node embeddings and structural representations)**.** *Let $\boldsymbol{Y}(\mathcal{S}, \mathbf{A}, \boldsymbol{X}) = (Y(\vec{S}, \mathbf{A}, \boldsymbol{X}))_{S\in\mathcal{S}}$ be as in Theorem 1. Consider a graph $G = (\mathbf{A}, \boldsymbol{X}) \in \Sigma$. Let $\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})$ be a most-expressive structural representation of nodes $S \in \mathcal{P}^\star(V)$ in G. Then,*

$$Y(\vec{S}, \mathbf{A}, \boldsymbol{X}) \perp\!\!\!\perp_{\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})} \boldsymbol{Z}|\mathbf{A}, \boldsymbol{X}, \quad \forall S \in \mathcal{S},$$

for any node embedding matrix $\boldsymbol{Z}$ that satisfies Definition 12, where $A \perp\!\!\!\perp_B C$ means A is independent of C given B. The above is a consequence of the fact that, for any embedding distribution $p(\boldsymbol{Z}|\mathbf{A}, \boldsymbol{X})$, there exists a $\mathcal{G}$-equivariant function ϕ such that $p(\boldsymbol{Z}|\mathbf{A}, \boldsymbol{X}) = \int_\epsilon \phi((\Gamma^\star(v, \mathbf{A}, \boldsymbol{X}))_{v\in V}, \epsilon) dP(\epsilon)$, where ϵ is a pure noise source and P its associated Lebesgue measure. Finally, $\forall(\mathbf{A}, \boldsymbol{X}) \in \Sigma$, there exists a most-expressive node embedding $\boldsymbol{Z}^\star|\mathbf{A}, \boldsymbol{X}$ such that,

$$\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X}) = \mathbb{E}_{\boldsymbol{Z}^\star}[f^{(|S|)}((\boldsymbol{Z}^\star_v)_{v\in S})|\mathbf{A}, \boldsymbol{X}], \quad \forall S \in \mathcal{S},$$

for some appropriate collection of functions $\{f^{(k)}(\cdot)\}_{k=1,\dots,n}$.

The proof of Theorem 2, given in the Appendix. It uses the fact that we can use the noise outsourcing theorem (Austin et al., 2008) to redefine $\boldsymbol{Z} \overset{\text{a.s.}}{=} \psi(\mathbf{A}, \boldsymbol{X}, \epsilon)$, where ψ is a $\mathcal{G}$-equivariant deterministic function of $\mathbf{A}$, $\boldsymbol{X}$, and a pure random noise ϵ that is independent of $\mathbf{A}$ and $\boldsymbol{X}$. The most-expressive embedding $\boldsymbol{Z}^\star|\mathbf{A}, \boldsymbol{X}$ extends the insight used to make GNNs more expressive in Murphy et al. (2019).

Theorem 2 implies that, *for any prediction task, node embeddings carry no information beyond that of structural representations.* A less attentive reader may think this creates an apparent paradox, since one cannot predict a property $Y((\textit{lynx,coyote}), \mathbf{A}_{\text{food web}}, \boldsymbol{X}_{\text{food web}})$ in Figure 1 from structural node embeddings, since $\Gamma(\textit{lynx}, \mathbf{A}, \boldsymbol{X}) = \Gamma(\textit{orca}, \mathbf{A}, \boldsymbol{X})$. The resolution of the paradox is to note that Theorem 2 describes the prediction of a link through a pairwise structural representation $\Gamma((\textit{lynx,coyote}), \mathbf{A}_{\text{food web}}, \boldsymbol{X}_{\text{food web}})$, and we may not be able to do the same task with structural node representations alone. An interesting question for future work is how well can we learn distributions (representations) from (node embeddings) samples (Kamath et al., 2015).

Other equally important consequences of Theorem 2 are: (a) any sampling approach obtaining node embeddings $\boldsymbol{Z}$ is valid as long as it is $\mathcal{G}$-equivariant (Definition 12) and isomorphic nodes have

the same marginal distributions (Lemma 2). (b) Interestingly, convex optimization methods for matrix factorization can be seen as variance-reduction techniques. (c) Methods that give unique node embeddings are provably incorrect —are not $\mathcal{G}$-equivariant— if the embedding of any two isomorphic nodes are different.

Remark 2 (Some GNN methods give node embeddings not structural representations)**.** *The random edges added by GraphSAGE (Hamilton et al., 2017a) and GIN (Xu et al., 2018) random walks make these methods node embeddings rather than structural node representations, according to Definition 12. To transform them back to structural node representations, one must average over all such random walks.*

The following corollaries describe other consequences of Theorem 2:

Corollary 2. *The link prediction task between any two nodes $u, v \in V$ depends only on the most-expressive tuple representation $\Gamma^\star((u,v), \mathbf{A}, \mathbf{X})$. Moreover, $\Gamma^\star((u,v), \mathbf{A}, \mathbf{X})$ always exists for any graph $(\mathbf{A}, \boldsymbol{X})$ and nodes (u,v). Finally, given most-expressive node embeddings $\boldsymbol{Z}^\star$, there exists a function f such that $\Gamma^\star((u,v), \mathbf{A}, \mathbf{X}) = \mathbb{E}_{\boldsymbol{Z}^\star}[f(\boldsymbol{Z}^\star_u, \boldsymbol{Z}^\star_v)]$, $\forall u, v$.*

A generalization of Corollary 2 is also possible, where Theorem 2 is used to allow us to create complex joint representations from simpler node embedding sampling methods.

Corollary 3. *Sample $\boldsymbol{Z}$ according to Definition 12. Then, we can learn a k-node structural representation of a subset of k nodes $S \in \mathcal{P}^\star(V)$, $|S| = k$, simply by learning a function $f^{(k)}$ whose average $\Gamma(\vec{S}, \mathbf{A}, \boldsymbol{X}) = \mathbb{E}[f^{(k)}((Z_v)_{v \in S})]$ can be used to predict $Y(\vec{S}, \mathbf{A}, \boldsymbol{X})$.*

The proof of Corollary 3 is in the Appendix. Finally, we show that the concepts of transductive and inductive learning are unrelated to the notions of node embeddings and structural representations.

Corollary 4. *Transductive and inductive learning are unrelated to the concepts of node embeddings and structural representations.*

Corollary 4 clears a confusion that, we believe, arises because traditional applications of node embeddings use a single Monte Carlo sample of $\boldsymbol{Z}|\mathbf{A}, \boldsymbol{X}$ to produce a structural representation (e.g., (Mikolov et al., 2013)). Inherently, a classifier learned with such a poor structural representation may fail to generalize over the test data, and will be deemed *transductive*.

Corollary 5. *A node embeddings sampling scheme can increase the structural representation power of GNNs.*

Corollary 5 is a direct consequence of the proof of Theorem 2, with Murphy et al. (2019) showing RP-GNN as a concrete method to do so.

4 RESULTS

This section focuses on applying the lessons learned in Section 3 in four tasks, divided into two common goals. The goal of the first three tasks is to show that, as described in Theorem 2, node embeddings can be used to create expressive structural embeddings of nodes, tuples, and triads. These representations are then subsequently used to make predictions on downstream tasks with varied node set sizes. The tasks also showcase the added value of using multiple node embeddings (Monte Carlo) samples to estimate structural representations, both during training and testing. Moreover, showcasing Theorem 1 and the inability of node representations to capture joint structural representations, these tasks show that structural node representations are useless in prediction tasks over more than one node, such as links and triads. The goal of fourth task is to showcase how multiple Monte Carlo samples of node embeddings are required to observe the fundamental relationship between structural representations and node embeddings predicted by Theorem 2.

An important note: Our proposed theoretical framework is not limited to the way we generate node embeddings. For example, our theoretical framework can use SVD in an inductive setting, where we train a classifier in one graph and test in a different graph, which was thought not possible previously with SVD. SVD with our theoretical framework is denoted MC-SVD, to emphasize the importance of Monte Carlo sampling in building better structural representations. Alternatively, more expressive node embeddings can be obtained, as we show next.

4.1 COLLIDING GRAPH NEURAL NETWORKS (CGNNS)

In this section we propose a new variational auto-encoder procedure to obtain node embeddings using neural networks, denoted Colliding Graph Neural Networks (CGNNs). Our sole reason to propose a new auto-encoder method is because we want to test the expressiveness of node embedding auto-encoders —and, unfortunately, existing auto-encoders, such as Grover et al. (2019), do not properly account for the dependencies introduced by the colliding variables in the graphical model. In our experiments, shown later, we aggregate multiple node embedding sampled from CGNN to obtain structural representations of the corresponding nodes and node sets.

Node Embedding Auto-encoder. In CGNN's, we adopt a latent variable approach to learn node embeddings. Corresponding to each evidence feature vector $\boldsymbol{X}_{i,\cdot} \in \mathbb{R}^k \; \forall \; i \in V$, we introduce a latent variable $\boldsymbol{Z}_{i,\cdot} \in \mathbb{R}^k$. In addition, our graphical model also consists of observed variables $\mathbf{A}_{i,j,\cdot} \in \mathbb{R}^d \; \forall \; i, j \in V$. These are related through the joint distribution $p(\mathbf{A}, \boldsymbol{X}|\boldsymbol{Z}) = \prod_{i,j \in V \times V} p(\mathbf{A}_{i,j,\cdot}|\boldsymbol{Z}_{i,\cdot}, \boldsymbol{Z}_{j,\cdot}) \prod_{h \in V} p(\boldsymbol{X}_{h,\cdot}|\boldsymbol{Z}_{h,\cdot})$, which is summarized by the Bayesian network in Figure 3 in the Appendix. Note that $\mathbf{A}_{i,j,\cdot}$ is a collider, since it is observed and influenced by two hidden variables, $\boldsymbol{Z}_{i,\cdot}, \boldsymbol{Z}_{j,\cdot}$. A neural network is used to learn the joint probability via MCMC, in an unsupervised fashion. The model learns the parameters of the MCMC transition kernel via an unrolled Gibbs Sampler, a templated recurrent model (an MLP with shared weights across Gibbs sampling steps), partially inspired by Fan & Huang (2017).

The unrolled Gibbs Sampler, starts with a normal distribution of the latent variables $\boldsymbol{Z}_{i,\cdot}^{(0)}, \forall i \in V$, with each $\boldsymbol{Z}_{i,\cdot}^{(0)} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$ independently, where I is the identity matrix. Subsequently at time steps $t = 1, 2, \ldots$, in accordance with the graphical model, each variable $\boldsymbol{Z}_{i,\cdot}^{(t)}$ is sequentially sampled from its true distribution conditioned on all observed edges of its corresponding node i, in addition to the most-up-to-date latent variables $\boldsymbol{Z}$'s associated with its immediate neighbors. The reparametrization trick (Kingma & Welling, 2013) allows us to backpropogate through the unrolled Gibbs Sampler. Algorithm 1 in the Appendix details our method. Consequently, this procedure has an effect on the run-time of this technique, which we alleviate by performing Parallel Gibbs Sampling by constructing parallel splashes (Gonzalez et al., 2011). Our unsupervised objective is to reconstruct the adjacency matrix.

4.2 QUANTITATIVE RESULTS

In what follows, we evaluate structural representations estimated from four node embedding techniques, namely GIN (Xu et al., 2018), RP-GIN (Murphy et al., 2019), MC-SVD and CGNN. We classify GIN and RP-GIN as node embedding techniques, as they employ the unsupervised learning procedure of Hamilton et al. (2017a). These were chosen because of their potential extra link and triad representation power over traditional structural representation GNNs. All node embedding methods are evaluated by their effect in estimating good structural representation for downstream task accuracy. We partition $G = (\mathbf{A}, \boldsymbol{X})$ into three non-overlapping induced subgraphs, namely $G_{\text{train}} = (\mathbf{A}_{\text{train}}, \boldsymbol{X}_{\text{train}})$, $G_{\text{val}} = (\mathbf{A}_{\text{val}}, \boldsymbol{X}_{\text{val}})$ and $G_{\text{test}} = (\mathbf{A}_{\text{test}}, \boldsymbol{X}_{\text{test}})$, which we use for training, validation and testing, respectively. In learning all four node embedding techniques, we only make use of the graphs G_{train} and G_{val}. All the four models used here have never seen the test graph G_{test} before test time —i.e., all our node embedding methods, used in the framework of Theorem 2, behave like inductive methods.

Monte Carlo joint representations during an **unsupervised** *learning phase:* A key component of our optimization is learning joint representations from node embeddings —as per Theorem 2. For this, at each gradient step (in practice, we do at each epoch), we perform a Monte Carlo sample of the node embeddings $\boldsymbol{Z}|\mathbf{A}, \boldsymbol{X}$. This, procedure optimizes a proper upper bound on the empirical loss, if the loss is the negative log-likelihood, cross-entropy, or a square loss. The proof is trivial by Jensen's inequality. For GIN and RP-GIN, we add random edges to the graph following a random walk at each epoch (Hamilton et al., 2017a). For the MC-SVD procedure, we use the left eigenvector matrix obtained by: (1) a random seed, (2) a random input permutation of the adjacency matrix, and (3) a single optimization step, rather than running SVD until it converges. We also have results with MC-SVD†, which is the same procedure as before, but runs SVD until convergence —noting that the latter is likely to give deterministic results in large real-world graphs.

Table 1: Micro F1 score on three distinct tasks averaged over 12 runs with standard deviation in parenthesis. The number within the parenthesis beside the model name indicates the number of Monte Carlo samples used in the estimation of the structural representation. MC-SVD$^{\dagger}(1)$ denotes the SVD procedure run until convergence with one Monte Carlo sample for the representation. Bold values show maximum empirical average, and multiple bolds happen when its standard deviation overlaps with another average. Results for Citeseer are provided in the Appendix in Table 2

	Node Classification			Link Prediction			Triad Prediction		
	Cora	Pubmed	PPI	Cora	Pubmed	PPI	Cora	Pubmed	PPI
Random	0.143	0.333	0.5^{121}	0.500	0.500	0.500	0.250	0.250	0.250
GIN(1)	0.646(0.021)	**0.878(0.006)**	0.533(0.003)	0.526(0.029)	0.513(0.048)	0.604(0.018)	0.280(0.010)	0.430(0.019)	0.400(0.006)
GIN(5)	0.676(0.031)	**0.880(0.003)**	0.535(0.004)	0.491(0.019)	0.517(0.028)	0.609(0.012)	0.284(0.017)	0.422(0.024)	0.397(0.004)
GIN(20)	0.678(0.024)	**0.880(0.002)**	0.536(0.003)	0.514(0.026)	0.512(0.042)	0.603(0.010)	0.281(0.010)	0.422(0.028)	0.399(0.004)
RP-GIN(1)	0.655(0.023)	**0.879(0.002)**	0.534(0.005)	0.506(0.016)	0.616(0.048)	0.605(0.011)	0.283(0.013)	0.423(0.024)	0.400(0.005)
RP-GIN(5)	0.681(0.022)	**0.881(0.004)**	0.534(0.004)	0.498(0.016)	0.637(0.038)	0.612(0.006)	0.285(0.025)	0.429(0.024)	0.399(0.009)
RP-GIN(20)	0.675(0.032)	**0.879(0.005)**	0.533(0.003)	0.518(0.017)	0.619(0.032)	0.603(0.007)	0.279(0.011)	0.418(0.011)	0.393(0.003)
MC-SVD$^{\dagger}(1)$	0.665(0.014)	0.810(0.009)	0.523(0.005)	0.588(0.029)	0.807(0.024)	**0.755(0.010)**	0.336(0.038)	0.515(0.077)	0.532(0.010)
MC-SVD(1)	0.667(0.017)	0.825(0.007)	0.521(0.006)	0.583(0.020)	0.818(0.032)	**0.755(0.008)**	0.304(0.034)	0.518(0.065)	0.529(0.006)
MC-SVD(5)	0.669(0.013)	0.842(0.015)	0.556(0.009)	0.572(0.019)	**0.848(0.038)**	**0.754(0.006)**	0.306(0.037)	**0.567(0.061)**	**0.544(0.008)**
MC-SVD(20)	0.672(0.013)	0.855(0.010)	0.591(0.009)	0.580(0.021)	**0.868(0.029)**	**0.762(0.010)**	0.300(0.033)	**0.546(0.029)**	**0.550(0.007)**
CGNN(1)	0.468(0.026)	0.686(0.020)	0.545(0.010)	0.682(0.026)	0.587(0.027)	0.661(0.015)	0.352(0.028)	0.404(0.014)	0.414(0.009)
CGNN(5)	0.641(0.022)	0.808(0.008)	0.637(0.014)	**0.707(0.027)**	0.585(0.037)	0.704(0.012)	**0.414(0.045)**	0.417(0.018)	0.463(0.026)
CGNN(20)	**0.726(0.024)**	0.831(0.010)	**0.707(0.015)**	**0.712(0.041)**	0.581(0.039)	0.738(0.011)	**0.405(0.034)**	0.419(0.017)	0.498(0.021)

Monte Carlo joint representations during a **supervised** *learning phase:* During the supervised phase, we first estimate a structural joint representation $\hat{\Gamma}(\vec{S}, \mathbf{A}, \boldsymbol{X})$ as the average of $m \in \{1, 5, 20\}$ Monte Carlo samples of a permutation-invariant function (Murphy et al., 2018; Zaheer et al., 2017) (sum-pooling followed by an MLP) applied to a sampled node embedding $(\boldsymbol{Z}_v)_{v \in S} | \mathbf{A}, \boldsymbol{X}$. Then, using $\hat{\Gamma}(\vec{S}, \mathbf{A}, \boldsymbol{X})$, we predict the corresponding target variable $Y(S, \mathbf{A}, \boldsymbol{X})$ of each task using an MLP. The node sets of our tasks $S \subseteq V$, have sizes $|S| \in \{1, 2, 3\}$, corresponding to node classification, link prediction, and triad prediction tasks, respectively.

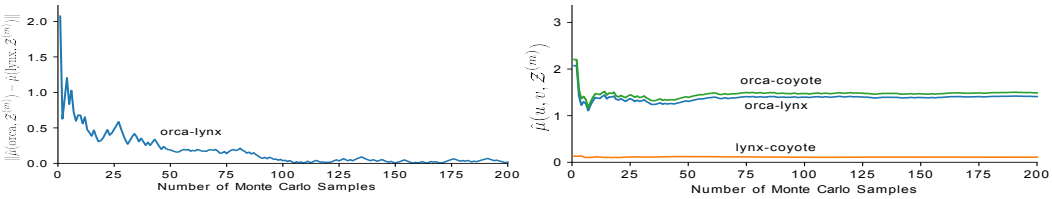
Datasets: We consider four graph datasets used by Hamilton et al. (2017a), namely Cora, Citeseer, Pubmed (Namata et al., 2012; Sen et al., 2008) and PPI (Zitnik & Leskovec, 2017). Cora, Citeseer and Pubmed are citation networks, where vertices represent papers, edges represent citations, and vertex features are bag-of-words representation of the document text. The PPI (protein-protein interaction) dataset is a collection of multiple graphs representing the human tissue, where vertices represent proteins, edges represent interactions across them, and node features include genetic and immunological information. Train, validation and test splits are used as proposed by Yang et al. (2016) (see Table 3 in the Appendix). Further dataset details can be found in the Appendix.

Node classification task: This task predicts node classes for each of the four datasets. In this task, structural node representations are enough. The structural node representation is used to classify nodes into different classes using an MLP, whose weights are trained in a supervised manner using the same splits as described above. In Cora, Citeseer and Pubmed, each vertex belongs only to a single class, whereas in the PPI graph dataset, nodes could belong to multiple classes.

Link prediction task: Here, we predict a small fraction of edges and non-edges in the test graph, as well as identify all false edges and non-edges (which were introduced as a corruption of the original graph) between different pairs of nodes in the graph. Specifically, we use joint tuple representations $\Gamma((u, v), \mathbf{A}, \boldsymbol{X})$, for $u, v \in V$, as prescribed by Theorem 2. Since, datasets are sparse in nature, and a trivial 'non-edges' predictor would result in a very high accuracy, we balance the train and validation and test splits to contain an equal number of edges and non-edges.

Triad prediction task: This task involves the prediction of triadic interaction as well as identification of possible fake interactions in the data between the three nodes under consideration. In this case, we use joint triadic representations $\Gamma((u, v, h), \mathbf{A}, \boldsymbol{X})$, for $u, v, h \in V$, as prescribed by Theorem 2. Here, we ensure that edge corruptions are dependent. We treat the graphs as being undirected in accordance with previous literature, and predict the number of true (uncorrupted) edges between the three nodes. Again, to handle the sparse nature of the graphs, we use a balanced dataset for train, validation, and test.

In Table 1 we present Micro-F1 scores for all four models over the three tasks. First, we note how more Monte Carlo samples at test time tend to increase test accuracy. In *node classification tasks*, we note that structural node representations from CGNN node embeddings significantly outperform other methods in two of the three datasets (the harder tasks). In *link prediction tasks*, the low accuracy of GNN-based methods (close to random) showcases the little extra-power of GIN and RP-GIN sampling schemes over the inability of their structural node representations to predict links.



(a) Difference in avg. structural node representations.

(b) Average structural link representations

Figure 2: Structural Representations for nodes and links using multiple samples obtained using MC-SVD on the disconnected food web graph shown in Figure 1.

Surprisingly, in *triads predictions*, the accuracy of GNN-based methods is much above random in some datasets, but still far from other node embedding methods. In *link and triad prediction tasks*, MC-SVD and CGNN share the lead with MC-SVD winning on Pubmed and PPI, and CGNN being significantly more accurate on Cora. Results for Citeseer, in the Appendix, show similar results.

4.3 Qualitative Results

We now investigate the transformation of node embedding into node and link structural representations. Theorem 2 shows that the average of a function over node embedding Monte Carlo samples gives node and link embedding. In this experiment, we empirically test Theorem 2, by creating structural representations from the node embedding random matrix $\boldsymbol{Z}$, defined as the left eigenvector matrix obtained through SVD (ran until convergence), with the sources of randomness being due to a random permutation of the adjacency matrix given as input to the SVD method and the random seed it uses. Consider m such embedding matrices Monte Carlo samples, $\mathcal{Z}^{(m)} = \{\boldsymbol{Z}^{(i)}\}_{i=1}^{m}$.

Structural node representations from node embeddings: According to Theorem 2, the average $\mathbb{E}[\boldsymbol{Z}_{v,\cdot}|\mathbf{A}]$ is a valid structural representation of node $v \in V$ in the adjacency matrix $\mathbf{A}$ of Figure 1. To test this empirically, we consider the unbiased estimator $\hat{\mu}(v, \mathcal{Z}^{(m)}) = \frac{1}{m}\sum_{i=1}^{m} \boldsymbol{Z}_{v,\cdot}^{(i)}$, $v \in V$, where $\lim_{m\to\infty} \hat{\mu}(v, \mathcal{Z}^{(m)}) \overset{a.s.}{=} \mathbb{E}[\boldsymbol{Z}_{v,\cdot}|\mathbf{A}]$. Figure 2a shows the Euclidean distance between the empirical structural representations $\hat{\mu}(\text{orca}, \mathcal{Z}^{(m)})$ and $\hat{\mu}(\text{lynx}, \mathcal{Z}^{(m)})$ as a function of $m \in [1, 200]$. As expected, because these two nodes are isomorphic, $\|\hat{\mu}(\text{orca}, \mathcal{Z}^{(m)}) - \hat{\mu}(\text{lynx}, \mathcal{Z}^{(m)})\| \to 0$ as m grows, with $m = 100$ giving reasonably accurate results.

Structural link representations from node embeddings: According to Theorem 2, the average $\mathbb{E}[f^{(2)}(\boldsymbol{Z}_{u,\cdot}, \boldsymbol{Z}_{v,\cdot})|\mathbf{A}]$ of a function $f^{(2)}$ is a valid structural representation of a link with nodes $u, v \in V$ in the adjacency matrix $\mathbf{A}$ of Figure 1. As an example, we use $f^{(2)}(a, b) = \|a - b\|$, and define the unbiased estimator $\hat{\mu}(u, v, \mathcal{Z}^{(m)}) = \frac{1}{m}\sum_{i=1}^{m} \|Z_u^{(i)} - Z_v^{(i)}\|, \forall u, v \in V$, where $\lim_{m\to\infty} \hat{\mu}(u, v, \mathcal{Z}^{(m)}) \overset{a.s.}{=} \mathbb{E}[\|\boldsymbol{Z}_{u,\cdot} - \boldsymbol{Z}_{v,\cdot}\||\mathbf{A}]$. Figure 2b shows the impact of increasing the number of Monte Carlo samples m over the empirical structural representation of links. We observe that although the empirical node representations of the orca and the lynx seem to converge to the same value, $\lim_{m\to\infty} \hat{\mu}(\text{orca}, \mathcal{Z}^{(m)}) = \lim_{m\to\infty} \hat{\mu}(\text{lynx}, \mathcal{Z}^{(m)})$, their empirical joint representations with the coyote converge to different values, $\lim_{m\to\infty} \hat{\mu}(\text{lynx, coyote}, \mathcal{Z}^{(m)}) \neq \lim_{m\to\infty} \hat{\mu}(\text{orca, coyote}, \mathcal{Z}^{(m)})$, as predicted by Theorem 2. Also note a similar (but weaker) trend for $\lim_{m\to\infty} \hat{\mu}(\text{orca, lynx}, \mathcal{Z}^{(m)}) \neq \lim_{m\to\infty} \hat{\mu}(\text{orca, coyote}, \mathcal{Z}^{(m)})$, showing these three tuples to be structurally different.

5 Related Work

Node Embeddings vs Structural Representations: Prior works have categorized themselves as one of either node embedding methods or methods which learn structural representations. This artificial separation, consequently led to little contemplation of the relation between the two, restricting each of these approaches to a certain subsets of downstream tasks on graphs. Node embeddings were arguably first defined in 1904, through Spearman's common factors. Ever since, there has never been a universal definition of node embedding: node embeddings were simply the product of a particular method. This literature features a myriad of methods, e.g., matrix factorization (Belkin & Niyogi, 2002; Cao et al., 2015; Ahmed et al., 2013; Ou et al., 2016), implicit matrix factorization (Mikolov et al., 2013; Arora et al., 2009; Chen et al., 2012; Perozzi et al., 2014; Grover & Leskovec,

2016), Bayesian factor models (Mnih & Salakhutdinov, 2008; Gopalan et al., 2014a), and some types of neural networks (You et al., 2019; Liang et al., 2018; Tang et al., 2019; Grover et al., 2019).

Arguably, the most common interpretation of node embeddings borrows from definitions of graph (node) embeddings in metric spaces: a measure of relative node *closeness* (Abraham et al., 2006; Bourgain, 1985; Candès & Recht, 2009; Graham & Winkler, 1985; Kleinberg, 2007; Linial et al., 1995; Rabinovich & Raz, 1998; Recht et al., 2010). Even in non-metric methods, such as word2vec (Mikolov et al., 2013) and Glove (Pennington et al., 2014), the embeddings have properties similar to those of metric spaces (Nematzadeh et al., 2017). Note that the definition of *close* varies from method to method, i.e., it is *model-dependent*. Still, this interpretation of *closeness* is the reason why their downstream tasks are often link prediction and clustering. However, once the literature started defining relative node closeness with respect to structural neighborhood similarities (e.g., Henderson et al. (2012); Ribeiro et al. (2017); Donnat et al. (2018)), node embeddings and structural representations became more strangely entangled.

Structural representations have an increasing body of literature focused on nodes and whole-graph classification tasks. Theoretically, these works abandon metric spaces in favor of a group-theoretic description of graphs (Bloem-Reddy & Teh, 2019; Chen et al., 2019; Kondor & Trivedi, 2018; Maron et al., 2018; Murphy et al., 2019), with connections to finite exchangeability and prior work on multilayer perceptrons (Wood & Shawe-Taylor, 1996). Graph neural networks (GNNs) (e.g., (Duvenaud et al., 2015; Gilmer et al., 2017; Kipf & Welling, 2016a; Hamilton et al., 2017a; Xu et al., 2018; Scarselli et al., 2008) among others) exploit this approach in tasks such as node and whole-graph classification. Morris et al. (2019) proposes a higher-order Weisfeller-Lehman GNN (WL-k-GNN), which is shown to get better accuracy in graph classification tasks than traditional (WL-1) GNNs. Unfortunately, Morris et al. (2019) focused only on graph-wide tasks, missing the fact that WL-2 GNN should be able to also perform link prediction tasks (Theorem 1), unlike WL-1 GNNs. GNN-like architectures have been used to simulate dynamic programming algorithms (Xu et al., 2019), which is unrelated to graphs and outside the scope of this work.

To the best of our knowledge, our work is the first to provide the theoretical foundations connecting node embeddings and structural representations. A few recent works have classified node embedding and graph representation methods arguing them to be fundamentally different (e.g., Hamilton et al. (2017b); Rossi et al. (2019); Wu et al. (2019); Zhou et al. (2018)). Rather, our work shows that these are actually equivalent for downstream classification tasks, with the difference being that one is a Monte Carlo method (embedding) and the other one is deterministic (representation).

Inductive vs Transductive Approaches: Another common misconception our work uncovers, is that of qualifying node embedding methods as transductive learning and graph representation ones as inductive (e.g., Hamilton et al. (2017a); Yang et al. (2016)). In their original definitions, transductive learning (Gammerman et al., 1998), (Zhu et al., 2003), (Zhou et al., 2004) and inductive learning (Michalski, 1983), (Belkin et al., 2006) are only to be distinguished on the basis of generalizability of the learned model to unobserved instances. The confusion seems to be rooted in researchers trying to use a single sample of a node embedding method and failing to generalize. Corollary 4 resolves this confusion by showing that transductive and inductive learning are fundamentally unrelated to node embeddings and graph representations. Both node embeddings and structural representations can be inductive if they can detect interesting conceptual patterns or reveal structure in the data. The theory provided by our work strongly adheres to this definition. Our work additionally provides the theoretical foundation behind the performance gains seen by Epasto & Perozzi (2019) and Goyal et al. (2019), which employ an ensemble of node embeddings for node classification tasks.

6 Conclusions

This work provided an invaluable unifying theoretical framework for node embeddings and structural graph representations, bridging methods like SVD and graph neural networks. Using invariant theory, we have shown (both theoretically and empirically) that relationship between structural representations and node embeddings is analogous to that of a distribution and its samples. We proved that all tasks that can be performed by node embeddings can also be performed by structural representations and vice-versa. Our empirical results show that node embeddings can be successfully used as inductive learning methods using our framework, and that non-GNN node embedding methods can be significantly more accurate in most tasks than a state-of-the-art GNNs method. Our work introduced new practical guidelines to the use of node embeddings, which we expect will replace today's naïve direct use of node embeddings in graph tasks.

REFERENCES

Ittai Abraham, Yair Bartal, and Ofer Neimany. Advances in metric embedding theory. In *Proceedings of the thirty-eighth annual ACM symposium on Theory of computing*, pp. 271–286. ACM, 2006.

Amr Ahmed, Nino Shervashidze, Shravan Narayanamurthy, Vanja Josifovski, and Alexander J Smola. Distributed large-scale natural graph factorization. In *Proceedings of the 22nd international conference on World Wide Web*, pp. 37–48. ACM, 2013.

Sanjeev Arora, Satish Rao, and Umesh Vazirani. Expander flows, geometric embeddings and graph partitioning. *Journal of the ACM (JACM)*, 56(2):5, 2009.

Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. A latent variable model approach to pmi-based word embeddings. *Transactions of the Association for Computational Linguistics*, 4:385–399, 2016.

Tim Austin et al. On exchangeable random variables and the statistics of large graphs and hypergraphs. *Probability Surveys*, 5:80–145, 2008.

Michael L Bates, Susan M Bengtson Nash, Darryl W Hawker, John Norbury, Jonny S Stark, and Roger A Cropp. Construction of a trophically complex near-shore antarctic food web model using the conservative normal framework with structural coexistence. *Journal of Marine Systems*, 145: 1–14, 2015.

Mikhail Belkin and Partha Niyogi. Laplacian eigenmaps and spectral techniques for embedding and clustering. In *Advances in neural information processing systems*, pp. 585–591, 2002.

Mikhail Belkin, Partha Niyogi, and Vikas Sindhwani. Manifold regularization: A geometric framework for learning from labeled and unlabeled examples. *Journal of machine learning research*, 7 (Nov):2399–2434, 2006.

Benjamin Bloem-Reddy and Yee Whye Teh. Probabilistic symmetry and invariant neural networks. *arXiv preprint arXiv:1901.06082*, 2019.

Jean Bourgain. On lipschitz embedding of finite metric spaces in hilbert space. *Israel Journal of Mathematics*, 52(1-2):46–52, 1985.

Emmanuel J Candès and Benjamin Recht. Exact matrix completion via convex optimization. *Foundations of Computational mathematics*, 9(6):717, 2009.

Shaosheng Cao, Wei Lu, and Qiongkai Xu. Grarep: Learning graph representations with global structural information. In *Proceedings of the 24th ACM international on conference on information and knowledge management*, pp. 891–900. ACM, 2015.

Shuo Chen, Josh L Moore, Douglas Turnbull, and Thorsten Joachims. Playlist prediction via metric embedding. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 714–722. ACM, 2012.

Zhengdao Chen, Soledad Villar, Lei Chen, and Joan Bruna. On the equivalence between graph isomorphism testing and function approximation with gnns. *NeruIPS*, 2019.

Thomas H Cormen, Charles E Leiserson, Ronald L Rivest, and Clifford Stein. *Introduction to algorithms*. MIT press, 2009.

Claire Donnat, Marinka Zitnik, David Hallac, and Jure Leskovec. Learning structural node embeddings via diffusion wavelets. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 1320–1329. ACM, 2018.

David K Duvenaud, Dougal Maclaurin, Jorge Iparraguirre, Rafael Bombarell, Timothy Hirzel, Alán Aspuru-Guzik, and Ryan P Adams. Convolutional networks on graphs for learning molecular fingerprints. In *Advances in neural information processing systems*, pp. 2224–2232, 2015.

Alessandro Epasto and Bryan Perozzi. Is a single embedding enough? learning node representations that capture multiple social contexts. In *The World Wide Web Conference*, pp. 394–404. ACM, 2019.

Shuangfei Fan and Bert Huang. Recurrent collective classification. *arXiv preprint arXiv:1703.06514*, 2017.

Alexander Gammerman, Volodya Vovk, and Vladimir Vapnik. Learning by transduction. In *Proceedings of the Fourteenth conference on Uncertainty in artificial intelligence*, pp. 148–155. Morgan Kaufmann Publishers Inc., 1998.

Justin Gilmer, Samuel S Schoenholz, Patrick F Riley, Oriol Vinyals, and George E Dahl. Neural message passing for quantum chemistry. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pp. 1263–1272. JMLR. org, 2017.

Joseph Gonzalez, Yucheng Low, Arthur Gretton, and Carlos Guestrin. Parallel gibbs sampling: From colored fields to thin junction trees. In *Proceedings of the Fourteenth International Conference on Artificial Intelligence and Statistics*, pp. 324–332, 2011.

Prem Gopalan, Francisco J Ruiz, Rajesh Ranganath, and David Blei. Bayesian nonparametric poisson factorization for recommendation systems. In *Artificial Intelligence and Statistics*, pp. 275–283, 2014a.

Prem K Gopalan, Laurent Charlin, and David Blei. Content-based recommendations with poisson factorization. In *Advances in Neural Information Processing Systems*, pp. 3176–3184, 2014b.

Palash Goyal, Di Huang, Sujit Rokka Chhetri, Arquimedes Canedo, Jaya Shree, and Evan Patterson. Graph representation ensemble learning. *arXiv preprint arXiv:1909.02811*, 2019.

Ronald L Graham and Peter M Winkler. On isometric embeddings of graphs. *Transactions of the American mathematical Society*, 288(2):527–536, 1985.

Aditya Grover and Jure Leskovec. node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 855–864. ACM, 2016.

Aditya Grover, Aaron Zweig, and Stefano Ermon. Graphite: Iterative generative modeling of graphs. *ICML*, 2019.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*, pp. 1024–1034, 2017a.

William L Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *arXiv preprint arXiv:1709.05584*, 2017b.

Keith Henderson, Brian Gallagher, Tina Eliassi-Rad, Hanghang Tong, Sugato Basu, Leman Akoglu, Danai Koutra, Christos Faloutsos, and Lei Li. Rolx: structural role extraction & mining in large graphs. In *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1231–1239. ACM, 2012.

Olav Kallenberg. *Foundations of modern probability*. Springer Science & Business Media, 2006.

Sudeep Kamath, Alon Orlitsky, Dheeraj Pichapati, and Ananda Theertha Suresh. On learning distributions from their samples. In *Conference on Learning Theory*, pp. 1066–1100, 2015.

Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Diederik P Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907*, 2016a.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. *arXiv preprint arXiv:1611.07308*, 2016b.

Robert Kleinberg. Geographic routing using hyperbolic space. In *IEEE INFOCOM 2007-26th IEEE International Conference on Computer Communications*, pp. 1902–1909. IEEE, 2007.

Risi Kondor and Shubhendu Trivedi. On the generalization of equivariance and convolution in neural networks to the action of compact groups. *ICML*, 2018.

Daniel D Lee and H Sebastian Seung. Algorithms for non-negative matrix factorization. In *Advances in neural information processing systems*, pp. 556–562, 2001.

Omer Levy and Yoav Goldberg. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*, pp. 2177–2185, 2014.

Dawen Liang, Rahul G Krishnan, Matthew D Hoffman, and Tony Jebara. Variational autoencoders for collaborative filtering. In *Proceedings of the 2018 World Wide Web Conference*, pp. 689–698. International World Wide Web Conferences Steering Committee, 2018.

Nathan Linial, Eran London, and Yuri Rabinovich. The geometry of graphs and some of its algorithmic applications. *Combinatorica*, 15(2):215–245, 1995.

Haggai Maron, Heli Ben-Hamu, Nadav Shamir, and Yaron Lipman. Invariant and equivariant graph networks. *arXiv preprint arXiv:1812.09902*, 2018.

Ryszard S Michalski. A theory and methodology of inductive learning. In *Machine learning*, pp. 83–134. Springer, 1983.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*, pp. 3111–3119, 2013.

Andriy Mnih and Ruslan R Salakhutdinov. Probabilistic matrix factorization. In *Advances in neural information processing systems*, pp. 1257–1264, 2008.

Christopher Morris, Martin Ritzert, Matthias Fey, William L Hamilton, Jan Eric Lenssen, Gaurav Rattan, and Martin Grohe. Weisfeiler and leman go neural: Higher-order graph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 33, pp. 4602–4609, 2019.

Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Janossy pooling: Learning deep permutation-invariant functions for variable-size inputs. *arXiv preprint arXiv:1811.01900*, 2018.

Ryan L Murphy, Balasubramaniam Srinivasan, Vinayak Rao, and Bruno Ribeiro. Relational pooling for graph representations. *arXiv preprint arXiv:1903.02541*, 2019.

Galileo Namata, Ben London, Lise Getoor, Bert Huang, and UMD EDU. Query-driven active surveying for collective classification. In *10th International Workshop on Mining and Learning with Graphs*, pp. 8, 2012.

Aida Nematzadeh, Stephan C Meylan, and Thomas L Griffiths. Evaluating vector-space models of word representation, or, the unreasonable effectiveness of counting words near other words. In *CogSci*, 2017.

Mingdong Ou, Peng Cui, Jian Pei, Ziwei Zhang, and Wenwu Zhu. Asymmetric transitivity preserving graph embedding. In *Proceedings of the 22nd ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 1105–1114. ACM, 2016.

Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pp. 1532–1543, 2014.

Bryan Perozzi, Rami Al-Rfou, and Steven Skiena. Deepwalk: Online learning of social representations. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pp. 701–710. ACM, 2014.

Yuri Rabinovich and Ran Raz. Lower bounds on the distortion of embedding finite metric spaces in graphs. *Discrete & Computational Geometry*, 19(1):79–94, 1998.

Benjamin Recht, Maryam Fazel, and Pablo A Parrilo. Guaranteed minimum-rank solutions of linear matrix equations via nuclear norm minimization. *SIAM review*, 52(3):471–501, 2010.

Leonardo FR Ribeiro, Pedro HP Saverese, and Daniel R Figueiredo. struc2vec: Learning node representations from structural identity. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, pp. 385–394. ACM, 2017.

Ryan A Rossi, Di Jin, Sungchul Kim, Nesreen K Ahmed, Danai Koutra, and John Boaz Lee. From community to role-based graph embeddings. *arXiv preprint arXiv:1908.08572*, 2019.

Franco Scarselli, Marco Gori, Ah Chung Tsoi, Markus Hagenbuchner, and Gabriele Monfardini. The graph neural network model. *IEEE Transactions on Neural Networks*, 20(1):61–80, 2008.

Prithviraj Sen, Galileo Namata, Mustafa Bilgic, Lise Getoor, Brian Galligher, and Tina Eliassi-Rad. Collective classification in network data. *AI magazine*, 29(3):93–93, 2008.

Charles Spearman. " general intelligence," objectively determined and measured. *The American Journal of Psychology*, 15(2):201–292, 1904.

Nils Chr Stenseth, Wilhelm Falck, Ottar N Bjørnstad, and Charles J Krebs. Population regulation in snowshoe hare and canadian lynx: asymmetric food web configurations between hare and lynx. *Proceedings of the National Academy of Sciences*, 94(10):5147–5152, 1997.

Da Tang, Dawen Liang, Tony Jebara, and Nicholas Ruozzi. Correlated variational auto-encoders. *arXiv preprint arXiv:1905.05335*, 2019.

Edward Wagstaff, Fabian B Fuchs, Martin Engelcke, Ingmar Posner, and Michael Osborne. On the limitations of representing functions on sets. *arXiv preprint arXiv:1901.09006*, 2019.

Jeffrey Wood and John Shawe-Taylor. A unifying framework for invariant pattern recognition. *Pattern Recognition Letters*, 17(14):1415–1422, 1996.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and Philip S Yu. A comprehensive survey on graph neural networks. *arXiv preprint arXiv:1901.00596*, 2019.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? *arXiv preprint arXiv:1810.00826*, 2018.

Keyulu Xu, Jingling Li, Mozhi Zhang, Simon S Du, Ken-ichi Kawarabayashi, and Stefanie Jegelka. What can neural networks reason about? *arXiv preprint arXiv:1905.13211*, 2019.

Zhilin Yang, William W Cohen, and Ruslan Salakhutdinov. Revisiting semi-supervised learning with graph embeddings. *ICML*, 2016.

Jiaxuan You, Rex Ying, and Jure Leskovec. Position-aware graph neural networks. *arXiv preprint arXiv:1906.04817*, 2019.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Advances in neural information processing systems*, pp. 3391–3401, 2017.

Dengyong Zhou, Olivier Bousquet, Thomas N Lal, Jason Weston, and Bernhard Schölkopf. Learning with local and global consistency. In *Advances in neural information processing systems*, pp. 321–328, 2004.

Jie Zhou, Ganqu Cui, Zhengyan Zhang, Cheng Yang, Zhiyuan Liu, and Maosong Sun. Graph neural networks: A review of methods and applications. *arXiv preprint arXiv:1812.08434*, 2018.

Xiaojin Zhu, Zoubin Ghahramani, and John D Lafferty. Semi-supervised learning using gaussian fields and harmonic functions. In *Proceedings of the 20th International conference on Machine learning (ICML-03)*, pp. 912–919, 2003.

Marinka Zitnik and Jure Leskovec. Predicting multicellular function through multi-layer tissue networks. *Bioinformatics*, 33(14):i190–i198, 2017.

APPENDIX

PRELIMINARIES

Noise outsourcing, representation learning, and graph representations: The description of our proofs starts with the equivalence between probability models of graphs and graph representations. We start with the concept of noise outsourcing (Austin et al., 2008, Lemma 3.1) applied to our task —a weaker version of the more general concept of transfer (Kallenberg, 2006, Theorem 6.10) in pushforward measures.

A probability law $\boldsymbol{Z}|(\mathbf{A}, \boldsymbol{X}) \sim p(\cdot|(\mathbf{A}, \boldsymbol{X}))$, $(\mathbf{A}, \boldsymbol{X}) \in \Sigma$, can be described (Kallenberg, 2006, Theorem 6.10) by pure random noise $\epsilon \sim \text{Uniform}(0, 1)$ independent of $(\mathbf{A}, \boldsymbol{X})$, passing through a deterministic function $\boldsymbol{Z} = f((\mathbf{A}, \boldsymbol{X}), \epsilon)$ —where $f : \Sigma \times [0, 1] \to \Omega$, where Ω in our task will be a matrix $\Omega = \mathbb{R}^{n \times d}$ defining node representations, $d \geq 1$. That is, the randomness in the conditional $p(z|(\mathbf{A}, \boldsymbol{X}))$ is entirely outsourced to ϵ, as f is deterministic.

Now, consider replacing the graph $G = (\mathbf{A}, \boldsymbol{X})$ by a $\mathcal{G}$-equivariant representation $\Gamma(\mathbf{A}, \boldsymbol{X})$ of its nodes, the output of a neural network $\Gamma : \Sigma \to \mathbb{R}^{n \times m}$, $m \geq 1$, that gives an representation to each node in G. If a representation $\Gamma^\star(\mathbf{A}, \boldsymbol{X})$ is such that $\exists f'$ where $\boldsymbol{Z} = f(\mathbf{A}, \boldsymbol{X}, \epsilon) = f'(\Gamma^\star(\mathbf{A}, \boldsymbol{X}), \epsilon)$, $\forall (\mathbf{A}, \boldsymbol{X}) \in \Sigma$ and $\forall \epsilon \in [0, 1]$, then $\Gamma^\star(\mathbf{A}, \boldsymbol{X})$ does not lose any information when it comes to predicting $\boldsymbol{Z}$. Statistically (Kallenberg, 2006, Proposition 6.13), $\boldsymbol{Z} \perp\!\!\!\perp_{\Gamma^\star(\mathbf{A}, \boldsymbol{X})} (\mathbf{A}, \boldsymbol{X})$. We call $\Gamma^\star(\mathbf{A}, \boldsymbol{X})$ a most-expressive representation of G with respect to $\boldsymbol{Z}$. A most-expressive representation (without qualifications) is one that is most-expressive for any target variable.

Representation learning is powerful precisely because it can learn functions Γ that are compact and can encode most of the information available on the input. And because the most-expressive $\Gamma^\star$ is $\mathcal{G}$-equivariant, it also guarantees that any $\mathcal{G}$-equivariant function over $\Gamma^\star$ that outputs $\boldsymbol{Z}$ is also $\mathcal{G}$-equivariant, without loss of information.

1 PROOF OF THEOREMS, LEMMAS AND COROLLARIES

We restate and prove **Lemma 1**

Lemma 1. *Two nodes $v, u \in V$, have the same most-expressive structural representation $\Gamma^\star(v, \mathbf{A}, \boldsymbol{X}) = \Gamma^\star(u, \mathbf{A}, \boldsymbol{X})$ iff u and v are isomorphic nodes in $G = (\mathbf{A}, \boldsymbol{X})$ (Definition 6).*

Proof. In this proof, we consider both directions.

($\Rightarrow$) Consider two nodes $v, u \in V$ which satisfy the condition, $\Gamma^\star(v, \mathbf{A}, \boldsymbol{X}) = \Gamma^\star(u, \mathbf{A}, \boldsymbol{X})$ but are not isomorphic in $G = (\mathbf{A}, \boldsymbol{X})$. By contradiction, suppose u and v have different node orbits. This is a contradiction, since the bijective mapping of Definition 9 would have to take the same input and map them to different outputs.

($\Leftarrow$) By contradiction, consider the two nodes $u, v \in V$ which are isomorphic in $G = (\mathbf{A}, \boldsymbol{X})$ but with different most expressive structural representations i.e. $\Gamma^\star(v, \mathbf{A}, \boldsymbol{X}) \neq \Gamma^\star(u, \mathbf{A}, \boldsymbol{X})$. This is a contradiction, because as per Definition 6 two nodes should have the same structural representation, which would imply the most expressive structural representation is not a structural representation. Hence, the two nodes should share the same most expressive structural representation. □

Next, we restate and prove **Lemma 2**

Lemma 2. *The permutation equivariance of p in Definition 12 implies that, if two proper subsets of nodes $S_1, S_2 \subsetneq V$ are isomorphic, then their marginal node embedding distributions must be the same up to a permutation, i.e., $p((\boldsymbol{Z}_i)_{i \in S_1}|\mathbf{A}, \boldsymbol{X}) = \pi(p((\boldsymbol{Z}_j)_{j \in S_2}|\mathbf{A}, \boldsymbol{X}))$ for some appropriate permutation π.*

Proof. From Definition 12, it is trivial to observe that two isomorphic nodes $u, v \in V$ in graph $G = (\mathbf{A}, \boldsymbol{X})$ have the same marginal node embedding distributions. In this proof we extend this to node sets $S \subset V$ where $|S| > 1$. We marginalize over $(Z_i)_{i \notin S_1}$ to obtain $p((\boldsymbol{Z}_i)_{i \in S_1}|\mathbf{A}, \boldsymbol{X})$ and in the other case over $(Z_i)_{i \notin S_2}$ to obtain $p((\boldsymbol{Z}_i)_{i \in S_2}|\mathbf{A}, \boldsymbol{X})$ respectively.

We consider 2 cases as follows:

Case 1: $S_1 = S_2$: This is the trivial where S_1 and S_2 are the exact same nodes, hence their marginal distributions are the identical as well by definition.

Case 2: $S_1 \neq S_2$: Since S_1 and S_2 are also given to be isomorphic, it is clear to see that every node in S_1 has an isomorphic equivalent in S_2. In a graph $G = (\mathbf{A}, \boldsymbol{X})$, the above statement conveys that S_2 can be written as a permutation π on S_1, i.e $S_2 = \pi(S_1)$. Now, employing Definition 12, it is clear to see that $p((\boldsymbol{Z}_i)_{i \in S_1} | \mathbf{A}, \boldsymbol{X}) = \pi(p((\boldsymbol{Z}_j)_{j \in S_2} | \mathbf{A}, \boldsymbol{X}))$ □

Next, we restate and prove **Theorem 1**

Theorem 1. *Let $\mathcal{S} \subseteq \mathcal{P}(V)$ be a set of subsets of V. Let $\boldsymbol{Y}(\mathcal{S}, \mathbf{A}, \boldsymbol{X}) = (Y(\vec{S}, \mathbf{A}, \boldsymbol{X}))_{S \in \mathcal{S}}$ be a sequence of random variables defined over the sets $S \in \mathcal{S}$ of a graph $G = (\mathbf{A}, \boldsymbol{X})$, such that we define $Y(\vec{S}, \mathbf{A}, \boldsymbol{X}) := Y_{\vec{S}} | \mathbf{A}, \boldsymbol{X}$ and $Y(\vec{S}_1, \mathbf{A}, \boldsymbol{X}) \overset{d}{=} Y(\vec{S}_2, \mathbf{A}, \boldsymbol{X})$ for any two jointly isomorphic subsets $S_1, S_2 \in \mathcal{S}$ in $(\mathbf{A}, \boldsymbol{X})$ (Definition 7), where $\overset{d}{=}$ means equality in their marginal distributions. Then, there exists a deterministic function φ such that, $\boldsymbol{Y}(\mathcal{S}, \mathbf{A}, \boldsymbol{X}) \overset{a.s.}{=} (\varphi(\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X}), \epsilon_S))_{S \in \mathcal{S}}$, where ϵ_S is a pure source of random noise from a joint distribution $p((\epsilon_{S'})_{\forall S' \in \mathcal{S}})$ independent of $\mathbf{A}$ and $\boldsymbol{X}$.*

Proof. The case $S = V$ is given in Theorem 12 of Bloem-Reddy & Teh (2019). The case $S = \emptyset$ is trivial. The case $S \subsetneq V$, $S \neq \emptyset$, is described as follows with a constructive proof. First consider the case of two isomorphic sets of nodes $S_1, S_2 \in \mathcal{S}$. As by definition $Y(\vec{S}_1, \mathbf{A}, \boldsymbol{X}) \overset{d}{=} Y(\vec{S}_2, \mathbf{A}, \boldsymbol{X})$, we must assume $p(Y_{\vec{S}_1} | \mathbf{A}, \boldsymbol{X}) = p(Y_{\vec{S}_2} | \mathbf{A}, \boldsymbol{X})$. We can now use the transfer theorem (Kallenberg, 2006, Theorem 6.10) to obtain a joint description $Y_{\vec{S}_1} | \mathbf{A}, \boldsymbol{X} \overset{\text{a.s.}}{=} \phi_1(\vec{S}_1, \mathbf{A}, \boldsymbol{X}, \epsilon)$ and $Y_{\vec{S}_2} | \mathbf{A}, \boldsymbol{X} \overset{\text{a.s.}}{=} \phi_2(\vec{S}_2, \mathbf{A}, \boldsymbol{X}, \epsilon)$, where ϵ is a common source of independent noise. As S_1 and S_2 are joint isomorphic (Definition 7), there exists an isomorphism $S_1 = \text{iso}(\vec{S}_2)$, where $\phi_1(\text{iso}(\vec{S}_2), \mathbf{A}, \boldsymbol{X}, \epsilon) = \phi_1(\vec{S}_1, \mathbf{A}, \boldsymbol{X}, \epsilon)$. Because the distribution given by $\phi_1(\cdot, \epsilon)$ must be isomorphic-invariant in $(\mathbf{A}, \boldsymbol{X})$ and S_1 and S_2 are also isomorphic in $(\mathbf{A}, \mathbf{X})$ then, for all permutation actions $\pi \in \Pi_n$, there exists a new isomorphism iso$'$ such that $\phi_1(\pi(\text{iso}(\vec{S}_2)), \pi(\mathbf{A}), \pi(\boldsymbol{X}), \epsilon) \overset{d}{=} \phi_1(\text{iso}'(\pi(\vec{S}_1)), \pi(\mathbf{A}), \pi(\boldsymbol{X}), \epsilon)$, which allows us to create a function φ' that incorporates iso$'$ into ϕ_1. Due to the isomorphism between S_1 and S_2, we can do the same process for S_2 to arrive at the same function φ'. We can now apply Corollary 6.11 (Kallenberg, 2006) over $(Y_{\vec{S}_1} | \mathbf{A}, \boldsymbol{X}, Y_{\vec{S}_2} | \mathbf{A}, \boldsymbol{X})$ along with a measure-preserving mapping f to show that $Y_{\vec{S}_1} | \mathbf{A}, \boldsymbol{X} \overset{a.s.}{=} \varphi'(\vec{S}_1, \mathbf{A}, \mathbf{X}, \epsilon_1)$ and $Y_{\vec{S}_2} | \mathbf{A}, \boldsymbol{X} \overset{a.s.}{=} \varphi'(\vec{S}_2, \mathbf{A}, \mathbf{X}, \epsilon_2)$, where $(\epsilon_1, \epsilon_2) = f(\epsilon)$. If S_1 and S_2 are not joint isomorphic, we can simply define $\varphi'(S_i, \cdot) := \phi_i(S_i, \cdot)$. Definition 11 allows us to define a function φ from which we rewrite $\varphi'(\vec{S}_i, \mathbf{A}, \mathbf{X}, \epsilon_i)$ as $\varphi(\Gamma^\star(\vec{S}_i, \mathbf{A}, \mathbf{X}), \epsilon_i)$. Applying the same procedure to all $S \in \mathcal{S}$ concludes our proof. □

Next, we restate and prove **Theorem 2**

Theorem 2 (The statistical equivalence between node embeddings and structural representations)**.** *Let $\boldsymbol{Y}(\mathcal{S}, \mathbf{A}, \boldsymbol{X}) = (Y(\vec{S}, \mathbf{A}, \boldsymbol{X}))_{S \in \mathcal{S}}$ be as in Theorem 1. Consider a graph $G = (\mathbf{A}, \boldsymbol{X}) \in \Sigma$. Let $\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})$ be a most-expressive structural representation of nodes $S \in \mathcal{S}$ in $(\mathbf{A}, \boldsymbol{X})$. Then,*

$$Y(\vec{S}, \mathbf{A}, \boldsymbol{X}) \perp\!\!\!\perp_{\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})} \boldsymbol{Z} | \mathbf{A}, \boldsymbol{X}, \quad \forall S \in \mathcal{S},$$

for any node embedding matrix $\boldsymbol{Z}$ that satisfies Definition 12, where $A \perp\!\!\!\perp_B C$ means A is independent of C given B. Finally, $\forall (\mathbf{A}, \boldsymbol{X}) \in \Sigma$, there exists a most-expressive node embedding $\boldsymbol{Z}^\star | \mathbf{A}, \boldsymbol{X}$ such that,

$$\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X}) = \mathbb{E}_{\boldsymbol{Z}^\star}[f^{(|S|)}((\boldsymbol{Z}^\star_v)_{v \in S}) | \mathbf{A}, \boldsymbol{X}], \quad \forall S \in \mathcal{S},$$

for some appropriate collection of functions $\{f^{(k)}(\cdot)\}_{k=1,\ldots,n}$.

Proof. In the first part of the proof, for any embedding distribution $p(\boldsymbol{Z} | \mathbf{A}, \boldsymbol{X})$, we note that the transfer theorem (Kallenberg, 2006, Theorem 6.10) implies that there exists an equivariant function ϕ such that $p(\boldsymbol{Z} | \mathbf{A}, \boldsymbol{X}) \overset{\text{a.s.}}{=} \phi((\Gamma^\star(v, \mathbf{A}, \boldsymbol{X}))_{v \in V}, \epsilon)$, where ϵ is a pure noise source. Then, the proof that $Y(\vec{S}, \mathbf{A}, \boldsymbol{X}) \perp\!\!\!\perp_{\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})} \boldsymbol{Z} | \mathbf{A}, \boldsymbol{X}$, $\forall S \in \mathcal{S}$, is a direct consequence of Proposition 6.13 in (Kallenberg, 2006).

In the second part of the proof, we construct an orbit over a most-expressive representation of a graph $(\mathbf{A}, \boldsymbol{X})$ of size n, with permutations that act only on unique node ids (node orderings) added as node features: $\Pi'(\mathbf{A}, \boldsymbol{X}) = \{((\Gamma^\star(v, \mathbf{A}, [\boldsymbol{X}, \pi(1, \ldots, n)^T]))_{\forall v \in V}\}_{\forall \pi \in \Pi_n}$, where $[A, b]$ concatenates column vector b as a column of matrix A. Define $\boldsymbol{Z}^\star | \mathbf{A}, \boldsymbol{X}$ as the random variable with a uniform measure over the set $\Pi'(\mathbf{A}, \boldsymbol{X})$. We first prove that $\boldsymbol{Z}^\star | \mathbf{A}, \boldsymbol{X}$ is a most-expressive node embedding. Clearly, $\boldsymbol{Z}^\star | \mathbf{A}, \boldsymbol{X}$ is a node embedding, since the uniform measure over $\Pi'(\mathbf{A}, \boldsymbol{X})$ is $\mathcal{G}$-equivariant. All that is left to show is that we can construct $\Gamma^\star$ of any-size subset $S \in \mathcal{S}$ from $\boldsymbol{Z}^\star | \mathbf{A}, \boldsymbol{X}$ via

$$\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X}) = \mathbb{E}_{\boldsymbol{Z}^\star}[f^{(|S|)}((\boldsymbol{Z}^\star_v)_{v \in S}) | \mathbf{A}, \boldsymbol{X}],$$

for some function $f^{(|S|)}$. This part of the proof has a constructive argument and comes in two parts.

Assume $S \in \mathcal{S}$ has no other joint isomorphic set of nodes in $\mathcal{S}$, i.e., $\nexists S_2 \in \mathcal{S}$ such that S and S_2 are joint isomorphic in $(\mathbf{A}, \boldsymbol{X})$. For any such subset of nodes $S \in \mathcal{S}$, and any element $R_\pi \in \Pi'(\mathbf{A}, \boldsymbol{X})$, there is a bijective measurable map between the nodes in S and their positions in the representation vector $R_\pi = (\Gamma^\star(v, \mathbf{A}, [\boldsymbol{X}, \pi(1, \ldots, n)^T]))_{\forall v \in V}$, since all node features are unique and there are no isomorphic nodes under such conditions. Consider the multiset

$$\mathcal{O}_S(\mathbf{A}, \boldsymbol{X}) := \{(\Gamma^\star(v, \mathbf{A}, [\boldsymbol{X}, \pi(1, \ldots, n)^T]))_{\forall v \in S}\}_{\forall \pi \in \Pi_n}$$

of the representations restricted to the set S. We now show that there exists an surjection between $\mathcal{O}_S(\mathbf{A}, \boldsymbol{X})$ and $\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})$. There is a surjection if for all $S_1, S_2 \in \mathcal{P}^\star(V)$ that are non-isomorphic, it implies $\mathcal{O}_{S_1}(\mathbf{A}, \boldsymbol{X}) \neq \mathcal{O}_{S_2}(\mathbf{A}, \boldsymbol{X})$. The condition is trivial if $|S_1| \neq |S_2|$ as $|\mathcal{O}_{S_1}(\mathbf{A}, \boldsymbol{X})| \neq |\mathcal{O}_{S_2}(\mathbf{A}, \boldsymbol{X})|$. If $|S_1| = |S_2|$, we prove by contradiction. Assume $\mathcal{O}_{S_1}(\mathbf{A}, \boldsymbol{X}) = \mathcal{O}_{S_2}(\mathbf{A}, \boldsymbol{X})$. Because of the unique feature ids and because $\Gamma^\star$ is most-expressive, the representation $\Gamma^\star(v, \mathbf{A}, [\boldsymbol{X}, \pi(1, \ldots, n)^T])$ of node $v \in V$ and permutation $\pi \in \Pi_n$ is unique. As S_1 is not isomorphic to S_2, and both sets have the same size, there must be at least one node $u \in S_1$ that has no isomorphic equivalent in S_2. Hence, there exists $\pi \in \Pi_n$ that gives a unique representation $\Gamma^\star(u, \mathbf{A}, [\boldsymbol{X}, \pi(1, \ldots, n)^T])$ that does not have a matching $\Gamma^\star(v, \mathbf{A}, [\boldsymbol{X}, \pi(1, \ldots, n)^T])$ for any $v \in S_2$ and $\pi' \in \Pi_n$. Therefore, $\exists a \in \mathcal{O}_{S_1}(\mathbf{A}, \boldsymbol{X})$, where $a \notin \mathcal{O}_{S_2}(\mathbf{A}, \boldsymbol{X})$, which is a contradiction since we assumed $\mathcal{O}_{S_1}(\mathbf{A}, \boldsymbol{X}) = \mathcal{O}_{S_2}(\mathbf{A}, \boldsymbol{X})$.

Now that we know there is such a surjection, a possible surjective measurable map between $\mathcal{O}_S(\mathbf{A}, \boldsymbol{X})$ and $\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})$ is a multiset function that takes $\mathcal{O}_S(\mathbf{A}, \boldsymbol{X})$ and outputs $\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})$. For finite multisets whose elements are real numbers $\mathbb{R}$, Wagstaff et al. (2019) shows that a most-expressive multiset function can be defined as the average of a function $f^{(|S|)}$ over the multiset. The elements of $\mathcal{O}_S(\mathbf{A}, \boldsymbol{X})$ are finite ordered sequences (ordered according to the permutation) and, thus, can be uniquely (bijectively) mapped to the real line with a measurable map, even when $\mathbf{A}$ and $\boldsymbol{X}$ have edge and node attributes defined over the real numbers $\mathbb{R}$. Thus, by Wagstaff et al. (2019), there exists some surjective function $f^{(|S|)}$ whose average over $\mathcal{O}_S(\mathbf{A}, \boldsymbol{X})$ give $\Gamma^\star(\vec{S}, \mathbf{A}, \boldsymbol{X})$.

Now assume $S_1, S_2 \subseteq V$ are joint isomorphic in $(\mathbf{A}, \boldsymbol{X})$, $S_1, S_2 \neq \emptyset$. Then, we have concluded that $\mathcal{O}_{S_1}(\mathbf{A}, \boldsymbol{X}) = \mathcal{O}_{S_2}(\mathbf{A}, \boldsymbol{X})$. Fortunately, by Definition 10, this non-uniqueness is a required property of the structural representations of $\Gamma^\star(S_1, \mathbf{A}, \boldsymbol{X})$ and $\Gamma^\star(S_2, \mathbf{A}, \boldsymbol{X})$, which must satisfy $\Gamma^\star(S_1, \mathbf{A}, \boldsymbol{X}) = \Gamma^\star(S_2, \mathbf{A}, \boldsymbol{X})$ if S_1 and S_2 are joint isomorphic, which concludes our proof. □

Next, we restate and prove **Corollary 1**

Corollary 1. *The node embeddings in Definition 12 encompass embeddings given by matrix and tensor factorization methods —such as Singular Value Decomposition (SVD), Non-negative Matrix Factorization (NMF), implicit matrix factorization (a.k.a. word2vec)–, latent embeddings given by Bayesian graph models —such as Probabilistic Matrix Factorizations (PMFs) and variants—, variational autoencoder methods and graph neural networks that use random lighthouses to extract node embedddings.*

Proof. In Probabilistic Matrix Factorization (Mnih & Salakhutdinov, 2008), we have $\mathbf{A}_{uv} \sim \mathcal{N}(Z_u^T Z_v, \sigma_a^2 \boldsymbol{I})$ where $Z_u \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$, $Z_v \sim \mathcal{N}(0, \sigma^2 \boldsymbol{I})$. We note that the posterior of $P(\boldsymbol{Z}|A)$ is clearly equivariant, satisfying definition 12, as a permutation action on the nodes requires the same permutation on the $\sigma^2 \boldsymbol{I}$ matrix as well to obtain $\boldsymbol{Z}$. The proof for Poisson Matrix Factorization (Gopalan et al., 2014a;b) follows a similar construction to the above, wherein the Normal assumption is replaced by the Poisson distribution.

Moreover, any matrix factorization algorithm gives an equivariant distribution of embeddings if the input matrices are randomly permuted upon input. Specifically, any Singular Value Decomposition (SVD) method satisfies Definition 12 as the distribution of the eigenvector solutions to degenerate singular values —which are invariant to unitary rotations in the corresponding degenerate eigenspace— will trivially be $\mathcal{G}$-equivariant even if the algorithm itself outputs values dependent on the node ids. Same is true for non-negative matrix factorization (Lee & Seung, 2001) and implicit matrix factorization (Levy & Goldberg, 2014; Mikolov et al., 2013).

PGNN's (You et al., 2019) compute the shortest distances between every node of the graph with a predetermined set of 'anchor' nodes to encode a distance metric. By definition, using such a distance metric would make the node embeddings learned by this technique $\mathcal{G}$-equivariant. The shortest path between all pairs of nodes in a graph can be seen equivalently as a function of a polynomial in $\mathbf{A}^k$. Alternatively, this can also be represented using the adjacency matrix and computed using the Floyd-Warshall algorithm (Cormen et al., 2009). The shortest distance is thus a function of $\mathbf{A}$ ignoring the node features $\boldsymbol{X}$. Since the inputs to the GNN comprises of the distance metric, $\mathbf{A}$ and $\boldsymbol{X}$, the node embeddings $\boldsymbol{Z}$ can equivalently seen as a function of $\mathbf{A}$, $\boldsymbol{X}$ and noise. The noise in this case is characterized by the randomized anchor set selection.

In variational auto-encoder models such as CVAE's, GVAE's, Graphite (Tang et al., 2019; Kipf & Welling, 2016b; Grover et al., 2019) the latent representations $\boldsymbol{Z}$ are learned either via a mean field approximation or are sampled independently of each other i.e. $\boldsymbol{Z} \sim P(\cdot|\mathbf{A}, \boldsymbol{X})$. We note that in the case of the mean field approximation, the probability distribution is a Dirac Delta. It is clear to see that the $\boldsymbol{Z}$ learned in this case is $\mathcal{G}$-equivariant with respect to any permutation action of the nodes in the graph.

□

Next, we restate and prove **Corollary 2**

Corollary 2. *The link prediction task between any two nodes $u, v \in V$ depends only on the most-expressive tuple representation $\Gamma^\star((u,v), \mathbf{A}, \mathbf{X})$. Moreover, $\Gamma^\star((u,v), \mathbf{A}, \mathbf{X})$ always exists for any graph $(\mathbf{A}, \boldsymbol{X})$ and nodes (u, v). Finally, given most-expressive node embeddings $\boldsymbol{Z}^\star$, there exists a function f such that $\Gamma^\star((u,v), \mathbf{A}, \mathbf{X}) = \mathbb{E}_{\boldsymbol{Z}^\star}[f(\boldsymbol{Z}^\star_u, \boldsymbol{Z}^\star_v)]$, $\forall u, v$.*

Proof. It is a consequence of Corollary 3 with $|S| = 2$. □

Next, we restate and prove **Corollary 3**

Corollary 3. *Sample $\boldsymbol{Z}$ according to Definition 12. Then, we can learn a k-node structural representation of a subset of k nodes $S \subseteq V$, $|S| = k$, simply by learning a function $f^{(k)}$ whose average $\Gamma(\vec{S}, \mathbf{A}, \boldsymbol{X}) = \mathbb{E}[f^{(k)}((Z_v)_{v \in S})]$ can be used to accurately predict $Y(\vec{S}, \mathbf{A}, \boldsymbol{X})$.*

Proof. This proof is a direct application of Theorem 2 which shows the statistical equivalence between node embeddings and strucutral representations.

Note that $f^{(k)}((\boldsymbol{Z}_v)_{v \in S})$ can equivalently be represented as $f^{(k)}(\varphi(\Gamma(v, \mathbf{A}, \boldsymbol{X})_{v \in S}, \epsilon_S))$ using Theorem 2 and that the noise ϵ_S is marginalized from the noise distribution of Theorem 1, still preserving equivariance. With an assumption of the most powerful $f'^{(k)}$, which is able to capture dependencies within the node set (Murphy et al., 2018) and noise ϵ_S, we can replace the above with $f'^{(k)}(\varphi(\Gamma(S, \mathbf{A}, \boldsymbol{X}), \epsilon_S))$ and subsequently compute an expectation over this function to eliminate the noise.

□

Next, we restate and prove **Corollary 4**

Corollary 4. *Transductive and inductive learning are unrelated to the concepts of node embeddings and structural representations.*

Proof. By Theorem 2, we can build most-expressive any-size joint representations from node embeddings, and we can get node embeddings from any-size most-expressive joint representations.

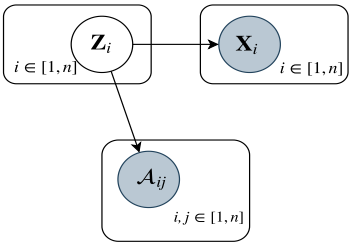


Figure 3: Latent variable model for Colliding Neural Networks . Observed evidence variables in gray

Hence, given enough computational resources, node embeddings and graph representations can have the same generalization performance over any tasks. This shows they are unrelated with the concepts of transductive and inductive learning. □

Next, we restate and prove **Corollary 5**

Corollary 5. *A node embeddings sampling scheme can increase the structural representation power of GNNs.*

Proof. The proof follows as a direct consequence of Theorem 2, along with Murphy et al. (2019) which demonstrates RP-GNN as a concrete method to do so. More specifically, appending unique node ids to node features uniformly at random, makes the nodes unique, and can be seen as a strategy to obtain node embeddings which satisfy Definition 12 using GNN's. By averaging over multiple such node embeddings gives us structural representations more powerful than that of standalone GNN's.

□

2 CGNN ALGORITHMS

The procedure to generate node embeddings used by the CGNN is given by Algorithm 1. Structural representations are computed as an unbiased estimate of the expected value of a function of the node embedding samples as given by Algorithm 2

input : $\mathbf{A}$, $\boldsymbol{X}$, *num-times*
output: $\boldsymbol{Z}$
initialization: $\boldsymbol{Z}_u \sim \mathcal{N}(0,1)\ \forall\ u \in V$
while *num-times* > 0 **do**
  for $u \in V$ **do**
    $\forall v \in V$ such that $\mathbf{A}_{uv} = 1$
    hidden $\leftarrow f(\{Z_v\})$; // f is a permutation invariant function
    visible $\leftarrow g(\{X_v\})$; // g is a permutation invariant function
    $Z_u \leftarrow$ **MLP** (hidden, visible, X_u) + **Noise** // With Reparametrization Trick
    // Equivalently, $\boldsymbol{Z_u} \sim P(\cdot|\{Z_v\},\{X_v\},\{A_{uv}\},X_u)$
  end
  num-times $\leftarrow$ num-times - 1
end

Algorithm 1: Node Embeddings from the Unrolled Gibbs Sampler

3 FURTHER RESULTS

In Table 2 we provide the results on node classification, link prediction and triad prediction on the Citeseer dataset.

input : $\{\mathbf{Z}^{(i)}\}_{i=1}^{m}$, k ;// node embedding samples, node set size
output: $g(\{\mathbf{Z}\}_{\boldsymbol{\mathcal{S}}})$; $\mathcal{S} = \{S_1\}_{\forall S_1 \subset V:|S_1|=k}$ //structural representations, $\mathbf{S}$ is a set of sets
initialization: $g(\{\mathbf{Z}\}_{\boldsymbol{\mathcal{S}}})$ = $\{\vec{0}\}$
for $i \in [1, m]$ **do**
  for $S \in \mathcal{S}$ **do**
    $g(\{Z_u\}_{u\in S}) \leftarrow g(\{Z_u\}_{u\in S})$ + $\frac{1}{m} f(\{Z_u\}_{u\in S})$ // f is a permutation invariant function
  end
end

Algorithm 2: Structural Representations from the Node Embedding Samples

Table 2: Micro F1 score on three distinct tasks over the Citeseer dataset, averaged over 12 runs with standard deviation in parenthesis. The number within the parenthesis beside the model name indicates the number of Monte Carlo samples used in the estimation of the structural representation. MC-SVD$^{\dagger}(1)$ denotes the SVD procedure run until convergence with one Monte Carlo sample for the representation. Bold values show maximum empirical average, and multiple bolds happen when its standard deviation overlaps with another average.

	Node Classification	Link Prediction	Triad Prediction
Random	0.167	0.500	0.250
GIN(1)	0.701(0.038)	0.543(0.024)	0.309(0.009)
GIN(5)	0.706(0.044)	0.525(0.040)	0.311(0.022)
GIN(20)	0.718(0.034)	0.530(0.023)	0.306(0.012)
RP-GIN(1)	0.719(0.031)	0.541(0.034)	0.313(0.005)
RP-GIN(5)	0.703(0.026)	0.539(0.025)	0.307(0.013)
RP-GIN(20)	0.724(0.020)	0.551(0.030)	0.307(0.017)
MC-SVD(1)	**0.734(0.007)**	0.562(0.017)	0.297(0.015)
MC-SVD(5)	**0.739(0.006)**	0.556(0.022)	0.302(0.009)
MC-SVD(20)	**0.737(0.005)**	0.565(0.020)	0.299(0.015)
CGNN(1)	0.689(0.010)	0.598(0.024)	0.305(0.009)
CGNN(5)	0.713(0.009)	0.627(0.048)	0.301(0.013)
CGNN(20)	0.721(0.008)	**0.654(0.049)**	0.296(0.008)

4 Description of Datasets and Experimental Setup

A detailed description of the datasets and the splits are given in Table 3. Our implementation is in PyTorch using Python 3.6. The implementations for GIN and RP-GIN are done using the PyTorch Geometric Framework. For GIN and RP-GIN, the embedding dimension was set to 256 at both convolutional layers. All MLPS, across all models have 256 neurons. Optimization is performed with the Adam Optimizer (Kingma & Ba, 2014). For the GIN, RP-GIN the learning rate was tuned in $\{0.01, 0.001, 0.0001, 0.00001\}$ whereas for CGNN's the learning rate was set to 0.001. Training was performed on Titan V GPU's. For more details refer to the code provided.

Table 3: Summary of the datasets

CHARACTERISTIC	**CORA**	**CITESEER**	**PUBMED**	**PPI**
Number of Vertices	2708	3327	19717	56944, 2373[a]
Number of Edges	10556	9104	88648	819994, 41000[a]
Number of Vertex Features	1433	3703	500	50
Number of Classes	7	6	3	121[b]
Number of Training Vertices	1208	1827	18217	44906[c]
Number of Validation Vertices	500	500	500	6514[c]
Number of Test Vertices	1000	1000	1000	5524[c]

[a] The PPI dataset comprises several graphs, so the quantities marked with an "a", represent the average characteristic of all graphs.
[b] For PPI, there are 121 targets, each taking values in $\{0, 1\}$.
[c] All of the training nodes come from 20 graphs while the validation and test nodes come from two graphs each not utilized during training.