TRACTABLE PROBABILISTIC GRAPH REPRESENTATION LEARNING WITH GRAPH-INDUCED SUM-PRODUCT NETWORKS

Federico Errica
NEC Laboratories Europe
Heidelberg, Germany

Mathias Niepert
University of Stuttgart
Stuttgart, Germany

ABSTRACT

We introduce Graph-Induced Sum-Product Networks (GSPNs), a new probabilistic framework for graph representation learning that can tractably answer probabilistic queries. Inspired by the computational trees induced by vertices in the context of message-passing neural networks, we build hierarchies of sum-product networks (SPNs) where the parameters of a parent SPN are learnable transformations of the a-posterior mixing probabilities of its children's sum units. Due to weight sharing and the tree-shaped computation graphs of GSPNs, we obtain the efficiency and efficacy of deep graph networks with the additional advantages of a probabilistic model. We show the model's competitiveness on scarce supervision scenarios, under missing data, and for graph classification in comparison to popular neural models. We complement the experiments with qualitative analyses on hyper-parameters and the model's ability to answer probabilistic queries.

1 INTRODUCTION

As the machine learning field advances towards highly effective models for language, vision, and applications in the sciences and engineering, many practical challenges stand in the way of widespread adoption. Overconfident predictions are hard to trust and most current models are not able to provide uncertainty estimates of their predictions (Pearl, 2009; Hüllermeier & Waegeman, 2021; Mena et al., 2021). Capturing such functionality requires the ability to efficiently answer probabilistic queries, e.g., computing the likelihood or marginals and, therefore, learning tractable distributions (Choi et al., 2020). Such capabilities would not only increase the trustworthiness of learning systems but also allow us to naturally cope with missing information in the input through marginalization, avoiding the use of ad-hoc imputation methods; this is another desideratum in applications where data (labels and attributes) is often incomplete (Dempster et al., 1977; Zio et al., 2007). In numerous application domains, obtaining labeled data is expensive, and while large unlabeled data sets might be available, this does not imply the availability of ground-truth labels. This is the case, for instance, in the medical domains (Tajbakhsh et al., 2020) where the labeling process must comply with privacy regulations and in the chemical domain where one gathers target labels via costly simulations or in-vitro experiments (Hao et al., 2020).

In this work, we are interested in data represented as a graph. Graphs are a useful representational paradigm in a large number of scientific disciplines such as chemistry and biology. The field of Graph Representation Learning (GRL) is concerned with the design of learning approaches that directly model the structural dependencies inherent in graphs (Bronstein et al., 2017; Hamilton et al., 2017b; Zhang et al., 2018b; Wu et al., 2020; Bacciu et al., 2020b). The majority of GRL methods implicitly induce a computational directed acyclic graph (DAG) for each vertex in the input graph, alternating learnable message passing and aggregation steps (Gilmer et al., 2017; Bacciu et al., 2020b). Most of these approaches exclusively rely on neural network components in these computation graphs, but cannot answer probabilistic queries nor exploit the vast amount of unlabeled data.

Motivated by these considerations, we propose a class of hierarchical probabilistic models for graphs called GSPNs, which can tractably answer a class of probabilistic queries of interest and whose computation graphs are also DAGs. While GSPNs are computationally as efficient as Deep Graph

Networks (DGNs), they consist of a hierarchy of interconnected Sum-Product Networks (SPNs) Poon & Domingos (2011). GSPNs can properly marginalize out missing data in graphs and answer probabilistic queries that indicate a change in likelihood under variations of vertex attribute values. The learned probabilistic graph representations are also competitive with state-of-the-art deep graph networks in the scarce supervision and graph classification settings. Overall, we provide evidence that it is possible to build a tractable family of SPNs to tackle learning problems defined on complex, non-Euclidean domains as compositions of simpler ones.

2 Related Work

Unsupervised learning for graphs is under-explored relative to the large body of work on supervised graph representation learning (Scarselli et al., 2009; Micheli, 2009; Niepert et al., 2016; Kipf & Welling, 2017; Velickovic et al., 2018; Xu et al., 2019; Ying et al., 2021). Contrary to self-supervised pre-training (Hu et al., 2020b) which investigates ad-hoc learning objectives, unsupervised learning encompasses a broader class of models that extract patterns from unlabeled data. Most unsupervised approaches for graphs currently rely on *i)* auto-encoders, such as the Graph Auto-Encoder (GAE) (Kipf & Welling, 2016) and *ii)* contrastive learning, with Deep Graph Infomax (DGI) (Velickovic et al., 2019) adapting ideas from computer vision and information theory to graph-structured data. While the former learns to reconstruct edges, the latter compares the input graph against its corrupted version and learns to produce different representations.

Existing probabilistic approaches to unsupervised deep learning on graphs often deal with clustering and Probability Density Estimation (PDE) problems; a classic example is the Gaussian Mixture Model (GMM) (Bishop, 2006) capturing multi-modal distributions of Euclidean data. The field of Statistical Relational Learning (SRL) considers domains where we require both uncertainty and complex relations; ideas from SRL have recently led to new variational frameworks for (un-)supervised vertex classification (Qu et al., 2019). Other works (Zheng et al., 2018) decompose the original graph into sub-graphs that are isomorphic to pre-defined templates to solve node-classification tasks. The Contextual Graph Markov Model (CGMM) (Bacciu et al., 2020a) and its variants (Atzeni et al., 2021; Castellana et al., 2022) are unsupervised DGNs trained incrementally, i.e., layer after layer, which have been successfully applied to graph classification tasks. Their incremental training grants closed-form learning equations at each layer, but it comes at the price of no global cooperation towards the optimization of the learning objective, i.e., the likelihood of the data; this makes it impractical for answering missing data queries.

So far, the literature on missing data in graphs, i.e., using the graph structure to handle missing attribute values, has not received as much attention as other problems. Recent work focuses on mitigating "missingness" in vertex classification tasks (Rossi et al., 2022) or representing missing information as a learned vector of parameters (Malone et al., 2021), but the *quality* of the learned data distribution has not been discussed so far (to the best of our knowledge). There are some attempts at imputing vertex attributes through a GMM (Taguchi et al., 2021), but such a process does not take into account the available graph structure. Similarly, other proposals (Chen et al., 2022) deal with the imputation of vertices having either all or none of their attributes missing, a rather unrealistic assumption for most cases. In this work, we propose an approach that captures the data distribution under missing vertex attribute values and requires no imputation.

3 Background

We define a graph g as a triple $(\mathcal{V}, \mathcal{E}, \mathcal{X})$, where $\mathcal{V} = \{1, \ldots, N\}$ denotes the set of N vertices, $\mathcal{E}$ is the set of *directed* edges (u, v) connecting vertex u to v, and $\mathcal{X} = \{\boldsymbol{x}_u \in \mathbb{R}^d, d \in \mathbb{N}, \forall u \in \mathcal{V}\}$ represents the the set of vertex attributes. When an edge is undirected, it is trasformed into two opposite directed edges (Bacciu et al., 2020b). In this work, we do not consider edge attributes. The *neighborhood* of a vertex v is the set $\mathcal{N}_v = \{u \in \mathcal{V} \mid (u, v) \in \mathcal{E}\}$. Also, access to the i-th component of a vector $\boldsymbol{x}$ shall be denoted by $\boldsymbol{x}(i)$ and that of a function f's output as $f(\cdot)_i$.

Graph Representation Learning Learning on graph-structured data typically means that one seeks a mapping from an input graph to vertex embeddings (Frasconi et al., 1998), and such a mapping should be able to deal with graphs of varying topology. In general, a vertex v's representation is

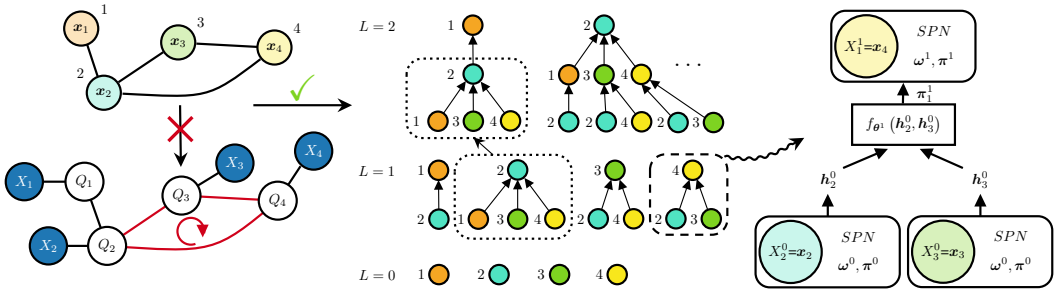

Figure 1: Inference on the graphical model on the left is typically unfeasible due to the mutual dependencies induced by cycles. Therefore, we approximate the learning problem using probabilistic computational trees of height L modeled by a hierarchy of tractable SPNs (right). Note that trees of height $L-1$ are used in the construction process of trees of height L. Also, for each tree rooted at v we visualize the mapping $m_v(\cdot)$ using colors and indices corresponding to the original graph (left).

a vector $\boldsymbol{h}_v \in \mathbb{R}^C, C \in \mathbb{N}$ encoding information about the vertex and its neighboring context, so that predictions about v can be then made by feeding $\boldsymbol{h}_v$ into a classical ML predictor. In graph classification or regression, instead, these vertex representations have to be globally aggregated using permutation invariant operators such as the sum or mean, resulting in a single representation $\boldsymbol{h}_g \in \mathbb{R}^C$ of the entire graph that is used by the subsequent predictor. At the time of this writing, message-passing neural networks (MPNN) (Gilmer et al., 2017) are the most popular class of methods to compute vertex representations. These methods adopt a local and iterative processing of information, in which each vertex repeatedly receives messages from its incoming connections, aggregates these messages, and sends new messages along the outgoing edges. Researchers have developed several variants of this scheme, starting from the two pioneering methods: the *recurrent* Graph Neural Network (GNN) of Scarselli et al. (2009) and the *feedforward* Neural Network for Graphs (NN4G) of Micheli (2009).

Tractable Probabilistic Models Probabilistic Circuits (PCs) are probabilistic models that can tractably, i.e., with *polynomial* complexity in the size of the circuit, answer a large class of probabilistic queries (Choi et al., 2020) and can be trained via backpropagation. A PC is usually composed of *distribution units*, representing distributions over one or more random variables, *product units* computing the fully factorized joint distribution of its children, and the *sum units* encoding mixtures of the children's distributions. Sum-Product Networks (SPNs) (Poon & Domingos, 2011; Gens & Domingos, 2012; Trapp et al., 2019; Shao et al., 2020) are a special class of PCs that support tractable computation of joint, marginal, and conditional distributions, which we will exploit to easily handle missing data. Informally, a (locally normalized) SPN is a probabilistic model defined via a rooted computational DAG, whose parameters of every sum unit add up to 1 (Vergari et al., 2019a). Furthermore, the *scope* of an SPN is the set of its distribution units, and *valid* SPNs represent proper distributions. For instance, a GMM and a Naïve Bayes model (NB) (Webb et al., 2010) can both be written as SPNs with a single sum unit. While the class of PCs is equivalent to that of deep mixture models, where each sum unit is associated with some latent variable (Peharz et al., 2016), PCs are *not* probabilistic graphical models (PGMs) (Koller & Friedman, 2009): the former specify an operational semantics describing how to compute probabilities, whereas the representational semantics of the latter specifies the conditional independence of the variables involved. Crucially, for valid SPNs and an arbitrary sum unit j with C children, it is always possible to tractably compute its posterior distribution, parametrized by the vector $\boldsymbol{h}_j \in \mathbb{R}^C$ (Peharz et al., 2016). Here, we are voluntarily abusing the notation because in the following we will consider posterior distributions of sum nodes as our latent representations. Due to space constraints, we provide a more detailed introduction to PCs in Section A.1.

4 GSPN: Learning Tractable Probabilistic Graph Representations

When one considers a cyclic graphical model such as the one on the left-hand-side of Figure 1, probabilistic inference is computationally infeasible unless we make specific assumptions to break

the mutual dependencies between the random variables (*r.v.*) Q, depicted as white circles. To address this issue, we approximate the intractable joint probability distribution over the *r.v.* attributed graph of any shape as products of tractable conditional distributions, one for each vertex in the graph; this approximation is known as the pseudo likelihood (Gong & Samaniego, 1981; Liang & Yu, 2003). The scope of these tractable distributions consists of the L-hop neighborhoods induced by a traversal of the graph rooted at said vertex. Akin to what done for DGNs, the parameters of the conditional distributions are shared across vertices.

We propose a class of models whose ability to tractably answer queries stems from a hierarchical composition of sum-product networks (SPNs). To describe the construction of this hierarchical model we consider, for each vertex v in the input graph, a tree rooted at v of height L, that is the length of the longest downward path to a leaf from v, treated as a hyper-parameter. Analogously, an internal node in the tree is said to have height $0 \leq \ell \leq L$. To distinguish between graphs and trees, we use the terms *vertices* for the graphs and *nodes* for the trees. Moreover, because graph cycles induce repetitions in the computational trees, we have to use a new indexing system: given a tree rooted at v with T_v nodes $\mathcal{T} = \{n_1, \ldots, n_{T_v}\}$, we denote by $m_v(\cdot) : \mathcal{T} \to \mathcal{V}$ a mapping from its node index n to a vertex u in the input graph g.
We now formally define the tree of height L rooted at vertex v as follows. First, we have one root node n with $m_v(n) = v$ and n having height L. Second, for a node n in the tree at height $1 \leq \ell \leq L$, we have that a node n' is a child of n with height $\ell - 1$ if and only if vertex $v' = m_v(n')$ is in the 1-hop neighborhood of vertex $u = m_v(n)$. When vertex u does not have incoming edges, we model it as a leaf node of the tree. Finally, every node n at height $\ell = 0$ is a leaf node and has no children. Figure 1 (center) depicts examples of trees induced by the nodes of the graph on the left.

We use the structure of each graph-induced tree as a blueprint to create a hierarchy of normalized SPNs, where all SPNs have the same internal structure. Every node of a tree is associated with a valid SPN whose scope consists of the random variables $A_1, \ldots, A_d$ modeling the vertex attributes, meaning the distribution units of said SPN are tractable distributions for the variable $\boldsymbol{X} = (A_1, ..., A_d)$. To distinguish between the various distributions for the *r.v.* $\boldsymbol{X}$ modeled by the SPNs at different nodes, we introduce for every node n and every height ℓ, the *r.v.* $\boldsymbol{X}_n^\ell$ with realization $\boldsymbol{x}_{m(n)}$. Crucially, these *r.v.* are fundamentally different even though they might have the same realization of the *r.v.* for root node . This is because each node of the tree encodes the contextual information of a vertex after a specific number of message passing steps.

The parameters of the tractable distribution units in the SPN of node n at height ℓ are denoted by $\boldsymbol{\omega}_n^\ell$, which are shared across the nodes at the same height. Moreover, the mixture probabilities of the S sum nodes in the SPN of node n at height ℓ are denoted by $\boldsymbol{\pi}_{n,j}^\ell$, $1 \leq j \leq S$. To obtain a hierarchical model, that is, to connect the SPNs according to the tree structure, we proceed as follows. For every SPN of a non-leaf node n, the parameters $\boldsymbol{\pi}_{n,j}^\ell$ of sum unit j are learnable transformations of the posterior probabilities of the (latent mixture component variables of) sum units j in the children SPNs. More formally, let $n_1, \ldots, n_T$ be the children of a node n and let $\boldsymbol{h}_{n_i,j}^{\ell-1}$ be the vector of posterior probabilities of sum unit j for the SPN of node n_i. Then, $\boldsymbol{\pi}_{n,j}^\ell = f_{\boldsymbol{\theta}_j^\ell}(\boldsymbol{h}_{n_1,j}^{\ell-1}, \ldots, \boldsymbol{h}_{n_T,j}^{\ell-1})$ with learnable parameters $\boldsymbol{\theta}_j^\ell$ shared across level ℓ. The choice of f does not depend on the specific SPN template, but it has to be permutation invariant because it acts as the neighborhood aggregation function of message-passing methods. The parameters of the sum units of leaf nodes, $\boldsymbol{\pi}_j^0$, are learnable and shared. Figure 1 (right) illustrates the hierarchical composition of SPNs according to a tree in Figure 1 (center). Moreover, Figure 2 illustrates how the prior distribution of the SPNs at height 1 is parametrized by a learnable transformation of posterior mixture probabilities of its children SPNs, similar to what is done by Vergari et al. (2019b) with fixed Dirichlet priors.

A graph g in the training data specifies (partial) evidence $\boldsymbol{x}_v$ for every one of its vertices v. The hierarchical SPN generated for v now defines a tractable probability distribution for the root node conditioned on the (partial) evidence for its intermediate nodes, which we can compute by evaluating the hierarchical SPNs tree in a bottom-up fashion. For each vertex v in the graph g, let $n_2, \ldots, n_{T_v}$ be the nodes of the graph-induced tree rooted at v, without the root node n_1 itself. For each graph g in the dataset, the objective is to maximize the following pseudo log-likelihood:

$$\log \prod_{v \in \mathcal{V}} P_{\boldsymbol{\Theta}}(\boldsymbol{x}_{m_v(n_1)} \mid \boldsymbol{x}_{m_v(n_2)}, \ldots, \boldsymbol{x}_{m_v(n_{T_v})}), \tag{1}$$

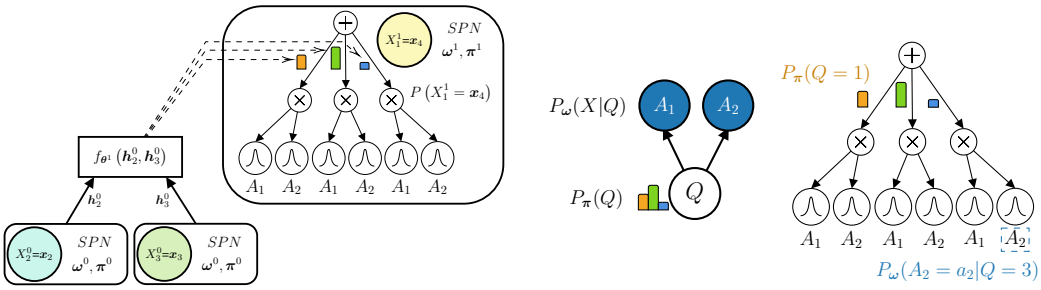

Figure 2: (Left) We expand the example of Figure 1 to illustrate how the prior distribution of the top SPN (here the Naïve Bayes on the right) is parametrized by a learnable transformation of the children's SPNs posterior mixture probabilities. (Right) A Gaussian Naïve Bayes graphical model with *r.v.* $\boldsymbol{X} = (A_1, A_2)$ and its equivalent SPN with scope $\{A_1, A_2\}$.

where $P_{\boldsymbol{\Theta}}$ is the conditional probability defined by the hierarchical SPN for the tree rooted at v with parameters $\boldsymbol{\Theta}$. When clear from the context, in the following we omit the subscript v from $m_v(\cdot)$.

4.1 Naive Bayes GSPNs

An instance of the proposed framework uses Naïve Bayes models as base SPNs, shown in Figure 2 in both their graphical and SPN representations for two continuous vertex attributes. For each SPN, there is a single categorical latent *r.v.* Q with C possible states. Let us denote this latent *r.v.* for node n at height ℓ in the tree as Q_n^ℓ and its prior distribution as $P_{\boldsymbol{\pi}_n^\ell}(Q_n^\ell)$. Due to the assumption that all *r.v.*s $\boldsymbol{X}_n^\ell$ have tractable distributions, we have, for all ℓ, n and i, that the conditional distribution $P_{\boldsymbol{\omega}_n^\ell}(\boldsymbol{X}_n^\ell \mid Q_n^\ell = i)$ is tractable. Moreover, for each child node n at height ℓ we have that

$$\boldsymbol{h}_n^\ell(i) = P(Q_n^\ell = i \mid \boldsymbol{X}_n^\ell = \boldsymbol{x}_{m(n)}). \tag{2}$$

For each leaf n of a tree, the posterior probabilities for the SPNs sum unit are given by

$$\boldsymbol{h}_n^0(i) = P(Q_n^0 = i \mid \boldsymbol{X}_n^0 = \boldsymbol{x}_{m(n)}) = \frac{P_{\boldsymbol{\omega}_n^0}(\boldsymbol{x}_{m(n)} \mid Q_n^0 = i) P_{\boldsymbol{\pi}_n^0}(Q_n^0 = i)}{\sum_{i'}^{C} P_{\boldsymbol{\omega}_n^0}(\boldsymbol{x}_{m(n)} \mid Q_n^0 = i') P_{\boldsymbol{\pi}_n^0}(Q_n^0 = i')}, \tag{3}$$

where $\boldsymbol{\pi}_n^0$ is learned and shared across leaf nodes. For $\ell \geq 1$, the latent priors $P_{\boldsymbol{\pi}_n^\ell}(Q_n^\ell)$ are parametrized by the output of a learnable transformation of the posterior probabilities of the sum units of the child SPNs. More formally, for node n' at height $\ell + 1, \ell \geq 0$, and with children $ch(n')$ we compute the prior probabilities $\boldsymbol{\pi}_{n'}^{\ell+1}$ as

$$\boldsymbol{\pi}_{n'}^{\ell+1} = f_{\boldsymbol{\theta}^{\ell+1}}(\boldsymbol{h}_1^\ell, ..., \boldsymbol{h}_{|ch_{n'}|}^\ell), \qquad \boldsymbol{h}_n^\ell(i) = \frac{P_{\boldsymbol{\omega}_n^\ell}(\boldsymbol{x}_{m(n)} \mid Q_n^\ell = i) P_{\boldsymbol{\pi}_n^\ell}(Q_n^\ell = i)}{\sum_{i'}^{C} P_{\boldsymbol{\omega}_n^\ell}(\boldsymbol{x}_{m(n)} \mid Q_n^\ell = i') P_{\boldsymbol{\pi}_n^\ell}(Q_n^\ell = i')}, \tag{4}$$

noting that the posterior $\boldsymbol{h}_n^\ell$ is tractable as the quantities of interest are obtained with a single backward pass in the SPN (Peharz et al., 2016). Figure 2 visualizes how $f_{\boldsymbol{\theta}^1}$ acts on the prior probabilities in the example of Figure 1. In the experiments, we define $f_{\boldsymbol{\theta}^{\ell+1}}(\boldsymbol{h}_1^\ell, ..., \boldsymbol{h}_{|ch_{n'}|}^\ell) = \frac{1}{|ch_{n'}|} \sum_{n \in ch_{n'}} \boldsymbol{\theta}^{\ell+1} \boldsymbol{h}_n^\ell$ for a learnable transition matrix $\boldsymbol{\theta}^{\ell+1} \in \mathbb{R}^{C \times C}$ that specifies how much a child's (soft) state k contributes to the weight of state i in the new prior distribution $P_{\boldsymbol{\pi}_{n'}^{\ell+1}}(Q_{n'}^{\ell+1} = i)$. Each row k of $\boldsymbol{\theta}^{\ell+1}$ must specify a valid probability over C possible states, enforced through normalization techniques. The function is motivated by the observation that it corresponds to applying the "Switching Parent" decomposition to the conditional mixture model defined in (Bacciu et al., 2020a) (deferred to Section A.2 due to space constraints). Likewise, we show in Section A.3 that Equation 4 produces a valid parametrization for the new prior distribution.

This iterative process can be efficiently implemented in the exact same way message passing is implemented in neural networks, because the computation for trees of height ℓ can be reused in trees of height $\ell + 1$. In practice, the height L of the computational tree corresponds to the number of layers in classical message passing. Furthermore, our choice of $f_{\boldsymbol{\theta}^\ell}$ leads to a fully probabilistic formulation and interpretation of end-to-end message passing on graphs. Motivated by studies on

the application of gradient ascent for mixture models and SPNs (Xu & Jordan, 1996; Sharir et al., 2016; Peharz et al., 2016; Gepperth & Pfülb, 2021), we maximize Equation 1 with backpropagation. We can execute all probabilistic operations on GPU to speed up the computation, whose overall complexity is linear in the number of edges as in most DGNs. For the interested reader, Section A.4 describes how GSPN can be applied to more general SPNs with multiple sum units and provides the pseudocode for the general inference phase; similarly, Section A.5 proposes a probabilistic shortcut mechanism akin to residual connections (Srivastava et al., 2015; He et al., 2016) in neural networks that mitigates the classical issues of training very deep networks via backpropagation. Finally, we refer to the unsupervised version of GSPN as GSPN_U.

4.2 Modeling Missing Data

With GSPNs we can deal with partial evidence of the graph, or missing attribute values, in a probabilistic manner. This is a distinctive characteristic of our proposal compared to previous probabilistic methods for vectors, as GSPNs can tractably answer probabilistic queries on graphs by also leveraging their structural information. Similarly, typical neural message passing models deal with missing data by imputing its values *before* processing the graph (Taguchi et al., 2021), whereas GSPN takes the partial evidence into account while processing the graph structure. We take inspiration from the EM algorithm for missing data (Hunt & Jorgensen, 2003): in particular, let us consider a multivariate *r.v.* of a root node $\boldsymbol{X}$ (dropping other indices for ease of notation) as a tuple of observed and missing sets of variables, i.e., $\boldsymbol{X} = (\boldsymbol{X}^{obs}, \boldsymbol{X}^{mis})$. When computing the posterior probabilities of sum nodes $\boldsymbol{h}$, we have to modify Equations 3 and 4 to only account for $\boldsymbol{X}^{obs}$; in SPNs, this equals to setting the distribution units of the missing attributes to 1 when computing marginals, causing the missing variables to be marginalized out. This can be computed efficiently due to the *decomposability* property of the SPNs used in this work (Darwiche, 2003). Additionally, if we wanted to impute missing attributes for a vertex v, we could apply the conditional mean imputation formula of Zio et al. (2007) to the corresponding root *r.v.* of the tree $\boldsymbol{X}_1^L = (\boldsymbol{X}_1^{obs}, \boldsymbol{X}_1^{mis})$, which, for the specific case of NB, translates to $\boldsymbol{x}_v^{mis} = \sum_i^C \mathbb{E}\big[\boldsymbol{X}_1^{mis} \mid Q_1^L = i\big] * \boldsymbol{h}_1^L(i)$. Hence, to impute the missing attributes of $\boldsymbol{x}_v$, we sum the (average) predictions of each mixture in the top SPN, where the mixing weights are given by the posterior distribution. The reason is that the posterior distribution carries information about our beliefs after observing $\boldsymbol{X}_v^{obs}$.

4.3 A Global Readout for Supervised Learning

Using similar arguments as before, we can build a probabilistic and supervised extension of GSPN for graph regression and classification tasks. It is sufficient to consider a new tree where the root node r is associated with the target *r.v.* Y and the children are all possible trees of different heights rooted at the N vertices of graph g. Then, we build an SPN for r whose sum unit j is parametrized by a learnable tansformation $\boldsymbol{\pi}_{r,j} = f_{\boldsymbol{\vartheta}_j}(\cdot)$. This function receives the set $\boldsymbol{h}_{\mathcal{V},j}$ of outputs $\boldsymbol{h}_{u,j}^{\ell}$ associated with the *top* SPN related to the graph-induced tree of height ℓ rooted at u, for $0 \leq \ell \leq L$. In other words, this is equivalent to consider all vertex representations computed by a DGN at different layers. For NB models with one sum unit, given (partial) evidence $\boldsymbol{x}_1, \ldots, \boldsymbol{x}_N$ for all nodes, we write

$$P(\boldsymbol{y} \mid \boldsymbol{x}_1, \ldots, \boldsymbol{x}_N) = \sum_{i=1}^{C_g} P_{\boldsymbol{\omega}_r}(\boldsymbol{y} \mid Q = i) P_{\boldsymbol{\pi}_r}(Q = i), \;\; \boldsymbol{\pi}_r \stackrel{\text{def}}{=} f_{\boldsymbol{\vartheta}}(\boldsymbol{h}_{\mathcal{V}}) = \Omega\big(\sum_{u \in \mathcal{V}} \sum_{\ell=1}^{L} \boldsymbol{\vartheta}^{\ell} \boldsymbol{h}_u^{\ell}\big), \tag{5}$$

where $\boldsymbol{\omega}_r$ and $\boldsymbol{\pi}_r$ are the parameters of the NB and Q is a latent categorical variable with C_g states. Computing $\boldsymbol{\pi}_r$ essentially corresponds to a global pooling operation, where $\boldsymbol{\vartheta}^{\ell} \in \mathbb{R}^{C_g \times C}$ is another transition matrix and Ω can be $\frac{1}{LN}$ (resp. the softmax function) for mean (resp. sum) global pooling. We treat the choice of $\Omega(\cdot)$ as a hyper-parameter, and the resulting model is called GSPN_S.

4.4 Limitations and Future Directions

Due to the problem of modeling an inherently intractable probability distribution defined over a cyclic graph, GSPNs rely on a composition of locally valid SPNs to tractably answer probabilistic queries. At training time the realization of the observable *r.v.* of vertex v at the root might be conditioned on the *same* realization of a *different* observable *r.v.* in the tree, which is meant to capture the mutual

dependency induced by a cycle. To avoid conditioning on the same realization, which could slightly bias the pseudo-likelihood in some corner cases, future work might consider an alternative, albeit expensive, training scheme where, for each node and observable attribute, a different computational tree is created and the internal nodes n with $m(n) = v$ are marginalized out. From the point of view of expressiveness in distinguishing non-isomorphic graphs, past results tell us that more powerful aggregations would be possible if we explored a different neighborhood aggregation scheme (Xu et al., 2019). In this work, we keep the neighborhood aggregation functions $f_{\boldsymbol{\theta}}$ as simple as possible to get a fully probabilistic model with valid and more interpretable probabilities. Finally, GSPN is currently unable to model edge types, which would enable us to capture a broader class of graphs.

5 EXPERIMENTS

We consider three different classes of experiments as described in the next sections. Code and data to reproduce the results is available at the following link: `https://github.com/nec-research/graph-sum-product-networks`.

Scarce Supervision Akin to Erhan et al. (2010) for non-structured data, we show that *unsupervised* learning can be very helpful in the scarce supervision scenario by exploiting large amounts of unlabeled data. We select seven chemical graph regression problems, namely *benzene*, *ethanol*, *naphthalene*, *salicylic acid*, *toluene*, *malonaldehyde* and *uracil* (Chmiela et al., 2017), using categorical atom types as attributes, and *ogbg-molpcba*, an imbalanced multi-label classification task (Hu et al., 2020a). Given the size of the datasets, we perform a hold-out risk assessment procedure (90% training, 10% test) with internal hold-out model selection (using 10% of the training data for validation) and a final re-training of the selected configuration. In the case of *ogbg-molpcba*, we use the publicly available data splits. Mean Average Error (MAE) is the main evaluation metric for all chemical tasks except for *ogbg-molpcba* (Average Precision – AP – averaged across tasks). We first train a GSPN_U on the whole training set to produce unsupervised vertex embeddings (obtained through a concatenation across layers), and then we fit a DeepSets (DS) (Zaheer et al., 2017) classifier on just 0.1% of the supervised samples using the learned embeddings (referred as GSPN_{U+DS}). We do not include GSPN_S in the analysis because, as expected, the scarce supervision led to training instability during preliminary experiments.
We compare against two effective and well-known unsupervised models, the Graph Auto-Encoder (GAE) and Deep Graph Infomax (DGI). These models have different unsupervised objectives: the former is trained to reconstruct the adjacency matrix whereas the second is trained with a contrastive learning criterion. Comparing GSPN_U, which maximizes the vertices' pseudo log-likelihood, against these baselines will shed light on the practical usefulness of our learning strategy. In addition, we consider the GIN model (Xu et al., 2019) trained exclusively on the labeled training samples as it is one of the most popular DGNs in the literature. We summarize the space of hyper-parameter configurations for all models in Table 4.

Modeling the Data Distribution under Partial Evidence We analyze GSPN's ability at modeling the missing data distribution on real-world datasets. For each vertex, we randomly mask a proportion of the attributes given by a sample from a Gamma distribution with concentration 1.5 and rate $1/2$). We do so to simulate a plausible scenario in which many vertices have few attributes missing and few vertices have many missing attributes. We train all models on the observed attributes only, and then we compute the negative log-likelihood of Equation 1 (NLL) for the masked attributes to understand how good each method is at modeling the missing data distribution. We apply the same evaluation, data split strategy, and datasets of the previous experiment, focusing on their 6 continuous attributes (except for *ogbg-molpcba* that has only categorical values).
The first baseline is a simple GAUSSIAN distribution, which computes its sufficient statistics (mean and standard deviation) from the training set and then computes the NLL on the dataset. The second baseline is a structure-agnostic Gaussian Mixture Model (GMM), which does not rely on the structure when performing imputation but can model arbitrarily complex distributions; we recall that our model behaves like a GMM when the number of layers is set to 1. The set of hyper-parameters configurations tried for each baseline is reported in Table 5.

Graph Classification Among the graph classification tasks of Errica et al. (2020), we restrict our evaluation to NCI1 (Wale et al., 2008), REDDIT-BINARY, REDDIT-MULTI-5K, and COLLAB

Model Eval. Process	GIN Sup.	GAE$_{+DS}$ Unsup.$\rightarrow$ Sup.	DGI$_{+DS}$ Unsup.$\rightarrow$ Sup.	GSPN$_{U+DS}$ Unsup.$\rightarrow$ Sup.
benzene	41.4 ± 45.6	$\mathbf{2.00} \pm 0.1$	6.22 ± 1.2	$\underline{2.24} \pm 0.5$
ethanol	4.00 ± 1.0	7.79 ± 7.8	5.35 ± 3.5	$\mathbf{3.36} \pm 0.0$
malonaldehyde	8.00 ± 5.3	5.49 ± 1.3	$\mathbf{4.31} \pm 1.8$	$\mathbf{4.31} \pm 1.4$
naphthalene	34.9 ± 19.4	4.45 ± 0.3	6.34 ± 2.3	$\mathbf{4.44} \pm 0.1$
salicylic acid	36.7 ± 13.4	233.3 ± 27.1	17.0 ± 7.3	$\mathbf{4.90} \pm 0.5$
toluene	29.6 ± 18.0	$\mathbf{4.83} \pm 0.3$	9.18 ± 0.3	$\underline{4.87} \pm 1.4$
uracil	19.7 ± 14.1	387.7 ± 13.7	409.7 ± 1.5	$\mathbf{3.93} \pm 0.0$
ogbg-molpcba ($\uparrow$)	$\mathbf{4.12} \pm 0.2$	3.36 ± 0.3	3.00 ± 0.5	$\underline{4.00} \pm 0.5$

Table 1: Mean and std results on scarce supervision tasks averaged over 10 runs, with 0.1%. The best and second-best average results are bold and underlined, respectively.

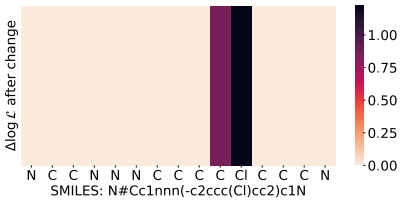

Figure 3: Relative change in vertices pseudo log-likelihood when replacing *Cl* in the SMILES with an *O*.

(Yanardag & Vishwanathan, 2015) datasets, for which leveraging the structure seems to help improving the performances. The empirical setup and data splits are the same as Errica et al. (2020) and Castellana et al. (2022), from which we report previous results. The baselines we compare against are a structure-agnostic baseline (BASELINE), DGCNN (Zhang et al., 2018a), DiffPool (Ying et al., 2018), ECC (Simonovsky & Komodakis, 2017), GIN (Xu et al., 2019), GraphSAGE (Hamilton et al., 2017a), CGMM, E-CGMM (Atzeni et al., 2021), and iCGMM (Castellana et al., 2022). CGMM can be seen as an incrementally trained version of GSPN that cannot use probabilistic shortcut connections. Table 6 shows the set of hyper-parameters tried for GSPN, and Table 9 shows that the time to compute a forward and backward pass of GSPN is comparable to GIN.

6 RESULTS

We present quantitative and qualitative results following the structure of Section 5. For all experiments we used a server with 32 cores, 128 GBs of RAM, and 8 Tesla-V100 GPUs with 32 GBs of memory.

Scarce Supervision In Table 1, we report the performance of GSPN$_U$ combined with DeepSets (DS) in the scarce supervision scenario. We make the following observations. First, a fully supervised model like GIN is rarely able to perform competitively when trained only on a small amount of labeled data. This is not true for *ogbg-molpcba*, but the AP scores are also very low, possibly because of the high class imbalance in the dataset. Secondly, on the first seven chemical tasks the unsupervised embeddings learned by GAE and DGI sometimes prevent DS from converging to a stable solution. This is the case for *ethanol*, *salicylic acid*, and *uracil*, where the corresponding MAE is higher than those of the GIN model. GSPN is competitive with and often outperforms the alternative approaches, both in terms of mean and standard deviation, and it ranks first or second place across all tasks. These results suggest that modeling the distribution of vertex attributes conditioned on the graph is a good inductive bias for learning meaningful vertex representations in the chemical domain and that exploiting a large amount of unsupervised graph data can be functional to solving downstream tasks in a scarce supervision scenario.

Contrarily to the other baselines, GSPN can answer probabilistic queries about the graph as depicted in Figure 3. Here we replace the Chlorine (*Cl*) atom in the SMILES of an *ogbg-molpcba* sample with an Oxygen (*O*) and ask the model to compute again the pseudo log-likelihood of each vertex (Equation 1). In this case, it makes intuitive sense that the relative likelihood increases because *Cl* is deactivating, so it is more unlikely to observe it attached to the all-carbon atom group. Being able to query the probabilistic model subject to changes in the input naturally confers a degree of interpretability and trustworthiness to the model. For the interested reader, we provide more visualizations for randomly picked samples in Section A.9.

Modeling the Data Distribution under Partial Evidence We demonstrate that is beneficial to model structural dependencies with GSPN to handle missing values in Table 2. The first observation is that modeling multimodality using a GMM already improves the NLL significantly over the unimodal GAUSSIAN baseline; though this may not seem surprising at first, we stress that this result is a good proxy to measure the usefulness of the selected datasets in terms of different possible values, and it establishes an upper bound for the NLL.

	benzene	ethanol	malonaldehyde	naphthalene	salicylic acid	toluene	uracil	S	M
GAUSSIAN	9.96 ± 0.0	5.72 ± 0.0	5.88 ± 0.0	6.48 ± 0.0	6.91 ± 0.0	6.00 ± 0.0	7.54 ± 0.0	✗	✗
GMM	4.31 ± 0.01	3.81 ± 0.01	3.93 ± 0.01	3.31 ± 0.10	3.21 ± 0.13	3.59 ± 0.06	$\mathbf{3.11} \pm 0.5$	✗	✓
GSPN$_U$	$\mathbf{4.17} \pm 0.02$	$\mathbf{3.77} \pm 0.03$	$\mathbf{3.88} \pm 0.01$	$\mathbf{3.18} \pm 0.07$	$\mathbf{3.06} \pm 0.21$	$\mathbf{3.35} \pm 0.05$	3.17 ± 0.17	✓	✓

Table 2: We report the mean and std NLL under the missing data scenario, averaged over 3 runs. Symbols S and M stand for "uses structure" and "captures multimodality", respectively.

	NCI1	**REDDIT-B**	**REDDIT-5K**	**COLLAB**
Baseline	69.8 ± 2.2	82.2 ± 3.0	52.2 ± 1.5	70.2 ± 1.5
DGCNN	76.4 ± 1.7	87.8 ± 2.5	49.2 ± 1.2	71.2 ± 1.9
DIFFPOOL	76.9 ± 1.9	89.1 ± 1.6	53.8 ± 1.4	68.9 ± 2.0
ECC	76.2 ± 1.4	-	-	-
GIN	$\mathbf{80.0} \pm 1.4$	89.9 ± 1.9	$\mathbf{56.1} \pm 1.7$	75.6 ± 2.3
GRAPHSAGE	76.0 ± 1.8	84.3 ± 1.9	50.0 ± 1.3	73.9 ± 1.7
CGMM$_{U+DS}$	76.2 ± 2.0	88.1 ± 1.9	52.4 ± 2.2	77.3 ± 2.2
E-CGMM$_{U+DS}$	$\underline{78.5} \pm 1.7$	89.5 ± 1.3	53.7 ± 1.0	77.5 ± 2.3
IC-GMM$_{U+DS}$	77.6 ± 1.5	$\mathbf{91.6} \pm 2.1$	$\underline{55.6} \pm 1.7$	$\mathbf{78.9} \pm 1.7$
GSPN$_{U+DS}$	$^{\dagger}76.6 \pm 1.9$	$^{\dagger}\underline{90.5} \pm 1.1$	$^{\dagger}55.3 \pm 2.0$	$^{\dagger}\underline{78.1} \pm 2.5$
GSPN$_S$	$^{\dagger}77.6 \pm 3.0$	$^{\dagger}89.7 \pm 2.3$	$^{\dagger}54.2 \pm 2.1$	74.1 ± 2.5

Table 3: Mean and std results on graph classification datasets. The best and second-best performances are bold and underlined, respectively. A † means that GSPN improves over CGMM.

Across almost all datasets GSPN improves the NLL score, and we attribute this positive result to its ability to simultaneously capture both the structural dependencies and the multimodality of the data distribution. The only dataset where the GMM has a better NLL is uracil, and we argue that this might be due to the independence of the node attributes from the surrounding structural information. These empirical results seem to agree with the considerations of Sections 4.2 and shed more light on a research direction worthy of further investigations.

Graph Classification The last quantitative analysis concerns graph classification, whose results are reported in Table 3. As we can see, not only is GSPN competitive against a consistent set of neural and probabilistic DGNs, but it also improves almost always w.r.t. $CGMM$, which can be seen as the layer-wise counterpart of our model. In addition, the model ranks second on two out of five tasks, though there is no clear winner among all models and the average performances are not statistically significant due to high variance. We also observe that here GSPN$_S$ does not bring significant performance gains compared to GSPN$_{U+DS}$. As mentioned in Section 4.4, we attribute this to the limited theoretical expressiveness of the global aggregation function. We also carry out a qualitative analysis on the impact of the number of layers, C and C_G for GSPN$_S$ (with global sum pooling) on the different datasets, to shed light on the benefits of different hyper-parameters. The results show that on NCI1 and COLLAB the performance improvement is consistent as we add more layers, whereas the opposite is true for the other two datasets where the performances decrease after five layers. Therefore, the selection of the best number of layers to use remains a non-trivial and dataset-specific challenge. In the interest of space, we report a visualization of these results in Appendix A.8.

7 CONCLUSIONS

We have proposed Graph-Induced Sum-Product Networks, a deep and fully probabilistic class of models that can tractably answer probabilistic queries on graph-structured data. GSPN is a composition of locally valid SPNs that mirrors the message-passing mechanism of DGNs, and it works as an unsupervised or supervised model depending on the problem at hand. We empirically demonstrated its efficacy across a diverse set of tasks, such as scarce supervision, modeling the data distribution under missing values, and graph prediction. Its probabilistic nature allows it to answer counterfactual queries on the graph, something that most DGNs cannot do. We hope our contribution will further bridge the gap between probabilistic and neural models for graph-structured data.

REFERENCES

Daniele Atzeni, Davide Bacciu, Federico Errica, and Alessio Micheli. Modeling edge features with deep bayesian graph networks. In *Proceedings of the International Joint Conference on Neural Networks (IJCNN)*, 2021.

Davide Bacciu, Alessio Micheli, and Alessandro Sperduti. Bottom-up generative modeling of tree-structured data. In *Proceedings of the 17th International Conference on Neural Information Processing (ICONIP)*, 2010.

Davide Bacciu, Federico Errica, and Alessio Micheli. Probabilistic learning on graphs via contextual architectures. *Journal of Machine Learning Research*, 21(134):1–39, 2020a.

Davide Bacciu, Federico Errica, Alessio Micheli, and Marco Podda. A gentle introduction to deep learning for graphs. *Neural Networks*, 129:203–221, 9 2020b.

Christopher M Bishop. *Pattern recognition and machine learning*. Springer, 2006.

Michael M. Bronstein, Joan Bruna, Yann LeCun, Arthur Szlam, and Pierre Vandergheynst. Geometric deep learning: going beyond Euclidean data. *IEEE Signal Processing Magazine*, 34(4):25. 18–42, 2017.

Daniele Castellana, Federico Errica, Davide Bacciu, and Alessio Micheli. The infinite contextual graph Markov model. In *Proceedings of the 39th International Conference on Machine Learning (ICML)*, 2022.

Xu Chen, Siheng Chen, Jiangchao Yao, Huangjie Zheng, Ya Zhang, and Ivor W. Tsang. Learning on Attribute-Missing Graphs. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44 (2):740–757, 2022.

Stefan Chmiela, Alexandre Tkatchenko, Huziel E Sauceda, Igor Poltavsky, Kristof T Schütt, and Klaus-Robert Müller. Machine learning of accurate energy-conserving molecular force fields. *Science advances*, 3(5):e1603015, 2017.

YooJung Choi, Antonio Vergari, and Guy Van den Broeck. Probabilistic circuits: A unifying framework for tractable probabilistic models. *Preprint*, 2020.

Adnan Darwiche. A differential approach to inference in bayesian networks. *Journal of the ACM (JACM)*, 50(3):280–305, 2003.

Arthur P Dempster, Nan M Laird, and Donald B Rubin. Maximum likelihood from incomplete data via the em algorithm. *Journal of the Royal Statistical Society: Series B (Methodological)*, 39(1): 1–22, 1977.

Dumitru Erhan, Aaron Courville, Yoshua Bengio, and Pascal Vincent. Why does unsupervised pre-training help deep learning? In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics (AISTATS)*, 2010.

Federico Errica, Marco Podda, Davide Bacciu, and Alessio Micheli. A fair comparison of graph neural networks for graph classification. In *8th International Conference on Learning Representations (ICLR)*, 2020.

Paolo Frasconi, Marco Gori, and Alessandro Sperduti. A general framework for adaptive processing of data structures. *IEEE Transactions on Neural Networks*, 9(5):768–786, 1998.

Robert Gens and Pedro Domingos. Discriminative learning of sum-product networks. *Proceedings of the 26th Conference on Neural Information Processing Systems (NeurIPS)*, 2012.

Alexander Gepperth and Benedikt Pfülb. Gradient-based training of gaussian mixture models for high-dimensional streaming data. *Neural Processing Letters*, 53(6):4331–4348, 2021.

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 (ICML)*, pp. 1263–1272, 2017.

Gail Gong and Francisco J Samaniego. Pseudo maximum likelihood estimation: theory and applications. *The Annals of Statistics*, pp. 861–869, 1981.

Will Hamilton, Zhitao Ying, and Jure Leskovec. Inductive representation learning on large graphs. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, 2017a.

William L. Hamilton, Rex Ying, and Jure Leskovec. Representation learning on graphs: Methods and applications. *IEEE Data Engineering Bulletin*, 40(3):52–74, 2017b.

Zhongkai Hao, Chengqiang Lu, Zhenya Huang, Hao Wang, Zheyuan Hu, Qi Liu, Enhong Chen, and Cheekong Lee. Asgn: An active semi-supervised graph neural network for molecular property prediction. In *Proceedings of the 26th International Conference on Knowledge Discovery and Data Mining (SIGKDD*, 2020.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the 29th IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.

Weihua Hu, Matthias Fey, Marinka Zitnik, Yuxiao Dong, Hongyu Ren, Bowen Liu, Michele Catasta, and Jure Leskovec. Open graph benchmark: Datasets for machine learning on graphs. In *Proceedings of the 34th Conference on Neural Information Processing Systems (NeurIPS)*, 2020a.

Weihua Hu, Bowen Liu, Joseph Gomes, Marinka Zitnik, Percy Liang, Vijay Pande, and Jure Leskovec. Strategies for pre-training graph neural networks. In *8th International Conference on Learning Representations (ICLR)*, 2020b.

Eyke Hüllermeier and Willem Waegeman. Aleatoric and epistemic uncertainty in machine learning: An introduction to concepts and methods. *Machine Learning*, 110(3):457–506, 2021.

Lynette Hunt and Murray Jorgensen. Mixture model clustering for mixed data with missing information. *Computational statistics & data analysis*, 41(3-4):429–440, 2003.

Thomas N Kipf and Max Welling. Variational graph auto-encoders. In *Workshop on Bayesian Deep Learning, Neural Information Processing System (NIPS)*, 2016.

Thomas N Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations (ICLR)*, 2017.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques*. MIT press, 2009.

Gang Liang and Bin Yu. Maximum pseudo likelihood estimation in network tomography. *IEEE Transactions on Signal Processing*, 51(8):2043–2053, 2003.

Brandon Malone, Alberto Garcia-Duran, and Mathias Niepert. Learning representations of missing data for predicting patient outcomes. In *Workshop on Deep Learning on Graphs: Method and Applications (AAAI)*, 2021.

José Mena, Oriol Pujol, and Jordi Vitrià. A survey on uncertainty estimation in deep learning classification systems from a bayesian perspective. *ACM Computing Surveys (CSUR)*, 54(9):1–35, 2021.

Alessio Micheli. Neural network for graphs: A contextual constructive approach. *IEEE Transactions on Neural Networks*, 20(3):498–511, 2009.

Mathias Niepert, Mohamed Ahmed, and Konstantin Kutzkov. Learning convolutional neural networks for graphs. In *Proceedings of the 33th International Conference on Machine Learning (ICML)*, 2016.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *IEEE transactions on pattern analysis and machine intelligence*, 39 (10):2030–2044, 2016.

Hoifung Poon and Pedro Domingos. Sum-product networks: A new deep architecture. In *IEEE International Conference on Computer Vision (ICCV), Workshops*, pp. 689–690. IEEE, 2011.

Meng Qu, Yoshua Bengio, and Jian Tang. GMNN: Graph Markov Neural Networks. In *Proceedings of the 36th International Conference on Machine Learning (ICML)*, 2019.

Emanuele Rossi, Henry Kenlay, Maria I. Gorinova, Benjamin Paul Chamberlain, Xiaowen Dong, and Michael M. Bronstein. On the unreasonable effectiveness of feature propagation in learning on graphs with missing node features. In *Proceedings of the 1st Learning on Graphs Conference (LoG)*, 2022.

Lawrence K Saul and Michael I Jordan. Mixed memory Markov models: Decomposing complex stochastic processes as mixtures of simpler ones. *Machine Learning*, 37(1):75–87, 1999.

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, 2009.

Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. Conditional sum-product networks: Imposing structure on deep probabilistic architectures. In *Proceedings of the 9th International Conference on Probabilistic Graphical Models (PGM)*, 2020.

Or Sharir, Ronen Tamari, Nadav Cohen, and Amnon Shashua. Tensorial mixture models. *arXiv preprint arXiv:1610.04167*, 2016.

Martin Simonovsky and Nikos Komodakis. Dynamic edge-conditioned filters in convolutional neural networks on graphs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2017.

Rupesh Kumar Srivastava, Klaus Greff, and Jürgen Schmidhuber. Highway networks. In *Deep Learning Workshop, 32nd International Conference on Machine Learning (ICML)*, 2015.

Hibiki Taguchi, Xin Liu, and Tsuyoshi Murata. Graph Convolutional Networks for Graphs Containing Missing Features. *Future Generation Computer Systems*, 117:155–168, April 2021. ISSN 0167739X. doi: 10.1016/j.future.2020.11.016. URL http://arxiv.org/abs/2007.04583.

Nima Tajbakhsh, Laura Jeyaseelan, Qian Li, Jeffrey N Chiang, Zhihao Wu, and Xiaowei Ding. Embracing imperfect datasets: A review of deep learning solutions for medical image segmentation. *Medical Image Analysis*, 63:101693, 2020.

Martin Trapp, Robert Peharz, Hong Ge, Franz Pernkopf, and Zoubin Ghahramani. Bayesian learning of sum-product networks. *Proceedings of the 33rd Conference on Neural Information Processing Systems (NeurIPS)*, 2019.

Petar Velickovic, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. Graph attention networks. In *6th International Conference on Learning Representations (ICLR)*, 2018.

Petar Velickovic, William Fedus, William L. Hamilton, Pietro Liò, Yoshua Bengio, and R. Devon Hjelm. Deep Graph Infomax. In *7th International Conference on Learning Representations (ICLR), New Orleans, LA, USA, May 6-9, 2019*, 2019.

Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Visualizing and understanding sum-product networks. *Machine Learning*, 108(4):551–573, 2019a.

Antonio Vergari, Alejandro Molina, Robert Peharz, Zoubin Ghahramani, Kristian Kersting, and Isabel Valera. Automatic bayesian density analysis. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence (AAAI)*, pp. 5207–5215, 2019b.

Nikil Wale, Ian A Watson, and George Karypis. Comparison of descriptor spaces for chemical compound retrieval and classification. *Knowledge and Information Systems*, 14(3):347–375, 2008.

Geoffrey I Webb, Eamonn Keogh, and Risto Miikkulainen. Naïve bayes. *Encyclopedia of machine learning*, 15:713–714, 2010.

Zonghan Wu, Shirui Pan, Fengwen Chen, Guodong Long, Chengqi Zhang, and S Yu Philip. A comprehensive survey on graph neural networks. *IEEE Transactions on Neural Networks and Learning Systems*, 2020.

Keyulu Xu, Weihua Hu, Jure Leskovec, and Stefanie Jegelka. How powerful are graph neural networks? In *7th International Conference on Learning Representations (ICLR)*, 2019.

Lei Xu and Michael I Jordan. On convergence properties of the em algorithm for gaussian mixtures. *Neural computation*, 8(1):129–151, 1996.

Pinar Yanardag and SVN Vishwanathan. Deep graph kernels. In *Proceedings of the 21th International Conference on Knowledge Discovery and Data Mining (SIGKDD*, 2015.

Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. Do transformers really perform badly for graph representation? In *Proceedings of the 35th Conference on Neural Information Processing Systems (NeurIPS)*, 2021.

Zhitao Ying, Jiaxuan You, Christopher Morris, Xiang Ren, Will Hamilton, and Jure Leskovec. Hierarchical graph representation learning with differentiable pooling. In *Proceedings of the 32nd Conference on Neural Information Processing Systems (NeurIPS)*, 2018.

Manzil Zaheer, Satwik Kottur, Siamak Ravanbakhsh, Barnabas Poczos, Ruslan R Salakhutdinov, and Alexander J Smola. Deep sets. In *Proceedings of the 31st Conference on Neural Information Processing Systems (NIPS)*, 2017.

Muhan Zhang, Zhicheng Cui, Marion Neumann, and Yixin Chen. An end-to-end deep learning architecture for graph classification. In *Proceedings of the 32nd AAAI Conference on Artificial Intelligence (AAAI)*, 2018a.

Ziwei Zhang, Peng Cui, and Wenwu Zhu. Deep learning on graphs: A survey. *arXiv preprint arXiv:1812.04202*, 2018b.

Kaiyu Zheng, Andrzej Pronobis, and Rajesh Rao. Learning graph-structured sum-product networks for probabilistic semantic maps. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Marco Di Zio, Ugo Guarnera, and Orietta Luzi. Imputation through finite gaussian mixture models. *Computational Statistics and Data Analysis*, 51(11):5305–5316, 2007. doi: 10.1016/j.csda.2006.10.002.