A APPENDIX

We complement the discussion with theoretical and practical considerations that facilitate understanding and reproducibility.

A.1 HIGH-LEVEL DEFINITIONS OF PROBABILISTIC CIRCUITS

We introduce basic concepts of PCs to ease the understanding of readers that are less familiar with this topic.

A probabilistic circuit (Choi et al., 2020) over a set of *r.v.*s $\boldsymbol{X} = (A_1, \ldots, A_d)$ is uniquely determined by its circuit structure and computes a (possibly unnormalized) distribution $P(\boldsymbol{X})$. The circuit structure has the form of a rooted DAG, which comprises a set of computational units. In particular, *input* units are those for which the set of incoming edges is empty (i.e., no children), whereas the *output* unit has no outgoing edges.

The *scope* of a PC is a function that associates each unit of the circuit with a subset of $\boldsymbol{X}$. For each non-input unit, the scope of the unit is the union of the scope of the children. It follows that the scope of the root is $\boldsymbol{X}$.

Each *input* (or *distribution*) unit encodes a parametric non-negative function, e.g., a Gaussian or Categorical distribution. A *product* unit represents the joint, fully factorized distribution between the distributions encoded by its children. Finally, a *sum* unit defines a weighted sum of the children's distributions, i.e., a mixture model when the children encode proper distributions. The set of parameters of a probabilistic circuit is given by the union of the parameters of all input and sum units in the circuit.

There are different kinds of probabilistic queries that a PC can answer. The first is the *complete evidence* query, corresponding to the computation of $P(\boldsymbol{X} = \boldsymbol{x})$. A single feedforward pass from the input units to the output unit is sufficient to compute this query. Another important class of queries is that of *marginals*, where we assume that not all *r.v.*s are fully observed, e.g., missing values. Given partial evidence $\boldsymbol{E} \subset \boldsymbol{X}$ and the unobserved variables $\boldsymbol{Z} = \boldsymbol{X} \setminus \boldsymbol{E}$, a marginal query is defined as $P(\boldsymbol{E} = \boldsymbol{e}) = \int p(\boldsymbol{e}, \boldsymbol{z}) d\boldsymbol{z}$. Finally, we mention the *conditional* query, sharing the same complexity as the marginal, where we compute the conditional probability $P(\boldsymbol{Q} = \boldsymbol{q} \mid \boldsymbol{E} = \boldsymbol{e})$ of a subset of *r.v.*s $\boldsymbol{Q} \subset \boldsymbol{X}$ conditioned on partial evidence $\boldsymbol{E}$ and $\boldsymbol{Z} = \boldsymbol{X} \setminus (\boldsymbol{E} \cup \boldsymbol{Q})$.

To be able to *tractably* compute the above queries, one usually wants the PC to have specific structural properties that guarantee a linear time complexity for marginal and conditional queries. A product unit is said to be *decomposable* when the scopes of its children are disjoint, and a PC is decomposable if all its product units are decomposable. Instead, a sum unit is *smooth* if all its children have identical scopes, and a PC is smooth (or *complete* (Poon & Domingos, 2011)) if all its sum units are smooth. For instance, the NB model considered in our work is implemented as a smooth and decomposable PC.

A PC that is smooth and decomposable can tractably compute marginals and conditional queries (one can prove that these are necessary and sufficient conditions for tractable computations of these queries), and we call such SPNs *valid*. A generalization of decomposability, namely *consistency*, is necessary to tractably compute maximum a posteriori queries of the form $\arg\max_{\boldsymbol{q}} P(\boldsymbol{Q} = \boldsymbol{q}, \boldsymbol{E} = \boldsymbol{e})$.

PCs have an interpretation in terms of latent variable models (Peharz et al., 2016), and in particular it is possible to augment PCs with specialized input units that mirror the latent variables of the associated graphical model. However, it is not immediate at all to see that *valid* SPNs allow a tractable computation of the posterior probabilities for the sum units, and we indeed refer the reader to works in the literature that formally prove it. As stated in (Poon & Domingos, 2011; Peharz et al., 2016), inference in unconstrained SPNs is generally intractable, but when an SPN is valid efficient inference is possible. Concretely, one can get all the required statistics for computing the posterior probabilities $\boldsymbol{h}_j$ of any sum unit j in a single backpropagation pass across the SPN (Equation 22 of Peharz et al. (2016)). The posterior computation involves the use of tractable quantities and hence stays tractable. Of course, the computational costs depend on the size of the SPN and might be impractical if the SPN is too large, but we still consider it tractable in terms of asymptotic time complexity.

A.2 THE SWITCHING PARENT DECOMPOSITION

The decomposition used in Equation 4, known as "Switching Parent" (SP) in the literature (Bacciu et al., 2010), was formally introduced in (Saul & Jordan, 1999) in the context of mixed memory Markov models. We report the original formulation below. Let $i_t \in \{1, \dots, n\}$ denote a discrete random variable that can take on n possible values. Then we write the SP decomposition as

$$P(i_t \mid i_{t-1}, \dots, i_{t-k}) = \sum_{\mu=1}^{n} P(\mu) P^{\mu}(i_t \mid i_{t-\mu}). \tag{6}$$

The connection to Equation 4 emerges by observing the following correspondences: $P(\mu)$ is treated as a constant $\frac{1}{|ch_n|}$, $P^{\mu}(i_t = i \mid i_{t-\mu} = j)$ implements the *transition* conditional probability table, which we model in a "soft" version (see also (Bacciu et al., 2020a)) as $\boldsymbol{\theta}^{\ell} \boldsymbol{h}_u^{\ell-1}$, and the conditional variables on the left-hand-side of the equation intuitively correspond to the information $\boldsymbol{h}_{ch_n}^{\ell-1}$ we use to parametrize the prior $P_{\boldsymbol{\pi}^{\ell}}(Q_v^{\ell})$. Moreover, we assume full stationarity and ignore the position of the child in the parametrization, meaning we use the same transition weights θ^{ℓ} for all neighbors; this is crucial since there is usually no consistent ordering of the vertices across different graphs, and consequently between the children in the SPN hierarchy.

A.3 PROOF THAT EQUATION 4 IS A VALID PARAMETRIZATION

To show that the computation $\boldsymbol{\pi}_{n'}^{\ell+1}$ outputs a valid parametrization for the categorical distribution $P_{\boldsymbol{\pi}_{n'}^{\ell+1}}(Q_{n'}^{\ell+1})$, it is sufficient to show that the parameters sum to 1:

$$\begin{aligned}
\sum_{i=1}^{C} \boldsymbol{\pi}_{n'}^{\ell+1}(i) = \sum_{i=1}^{C} f_{\boldsymbol{\theta}^{\ell+1}}(\boldsymbol{h}_1^{\ell}, ..., \boldsymbol{h}_{|ch_{n'}|}^{\ell})_i &= \sum_{i=1}^{C} \frac{1}{|ch_{n'}|} \sum_{n \in ch_{n'}} \sum_{k=1}^{C} \boldsymbol{\theta}_{ki}^{\ell+1} \boldsymbol{h}_n^{\ell}(k) \\
&= \frac{1}{|ch_{n'}|} \sum_{n \in ch_{n'}} \sum_{k=1}^{C} \sum_{i=1}^{C} \boldsymbol{\theta}_{ki}^{\ell+1} \boldsymbol{h}_n^{\ell}(k) \\
&= \frac{1}{|ch_{n'}|} \sum_{n \in ch_{n'}} \sum_{k=1}^{C} \boldsymbol{h}_n^{\ell}(k) = \frac{1}{|ch_{n'}|} \sum_{n \in ch_{n'}} 1 = 1,
\end{aligned} \tag{7}$$

where we used the fact that the rows of $\boldsymbol{\theta}^{\ell}$ and the posterior weights $\boldsymbol{h}_u^{\ell-1}$ are normalized.

A.4 DEALING WITH MORE GENERAL SPN TEMPLATES

In Section 4 we have introduced the general framework of GSPN for arbitrary SPNs, but the explicit implementation of $\boldsymbol{\pi}^{\ell}_{n,j} = f_{\boldsymbol{\theta}^{\ell}_j}(\boldsymbol{h}^{\ell-1}_{n_1,j}, \dots, \boldsymbol{h}^{\ell-1}_{n_T,j})$ and the computation of the posterior probabilities $\boldsymbol{h}^{\ell-1}_{n_i,j}$ has only been shown for the Naïve Bayes model (Section 4.1) with a *single* sum unit j in its corresponding SPN template. Despite that the computation of the posterior depends on the specific template used, we can still provide guidelines on how to use GSPN in the general case.

Consider any *valid* SPN template with S sum units, and *w.l.o.g.* we can assume that all sum units have C different weights, i.e., each sum unit implements a mixture of C distributions. Given a computational tree, we consider an internal node n with T children $n_1, \dots, n_T$, and we recall that all SPNs associated with the nodes of the tree share the same template (although a different parametrization). Therefore, there is a one-to-one correspondence between unit j of node n_i, $\forall i \in \{1, \dots, T\}$, at level $\ell - 1$ and unit j of its parent n at level ℓ, meaning that the parametrization $\boldsymbol{\pi}^{\ell}_{n,j} \in \mathbb{R}^C$ can be computed using a permutation invariant function such as

$$\boldsymbol{\pi}^{\ell}_{n,j} = f_{\boldsymbol{\theta}^{\ell}_j}(\boldsymbol{h}^{\ell-1}_{n_1,j}, \dots, \boldsymbol{h}^{\ell-1}_{n_T,j}) = \frac{1}{T}\sum_{i=1}^{T} \boldsymbol{\theta}^{\ell}_j \boldsymbol{h}^{\ell-1}_{n_i,j} \quad \forall j \in \{1, \dots, S\} \tag{8}$$

that acts similarly to the neighborhood aggregation function of DGNs.

All that remains is to describe how we compute the posterior probabilities $\boldsymbol{h}^{\ell}_{n,j} \in \mathbb{R}^C$ for all sum units $j \in \{1, \dots, S\}$ of a generic node n at level ℓ (now that the reader is familiar with the notation, we can abstract from the superscript ℓ since it can be determined from n). As discussed in Section A.1, SPNs have an interpretation in terms of latent variable models, so we can think of a sum unit j as a latent random variable and we can augment the SPN to make this connection explicit (Peharz et al., 2016). Whenever the SPN is valid, the computation of the posterior probabilities of all sum units is *tractable* and it requires *just one* backpropagation pass across the augmented SPN (see Equation 22 of Peharz et al. (2016)). This makes it possible to tractably compute the posterior probabilities of the sum units that will be used to parametrize the corresponding sum units of the parent node in the computational tree. Below we provide a pseudocode summarizing the inference process for a generic GSPN, but we remind the reader that the message passing procedure is equivalent to that of DGNs.

Algorithm 1 GSPN Inference on a Single Computational Tree

Input: Computational tree with T nodes and height L, valid SPN template with S sum units and C weights for each sum unit.
Output: Set of posterior probabilities $\{\boldsymbol{h}_{n,j} \ \ \forall n \in \{1, \dots, T\}, \forall j \in \{1, \dots, S\}\}$

for $\ell = 0, \dots, L$ **do**
  for all nodes n at height ℓ **do**
    for all sum units j of SPN of node n **do**:
      if $\ell > 0$ **then**:
        Compute $\boldsymbol{\pi}_{n,j}$ (Eq. 8) ▷ message passing, parallelized
      else
        Use learned $\boldsymbol{\pi}_{n,j}$ ▷ leaf node, no children
      end if
      Compute and collect $\boldsymbol{h}_{n,j}$ (Equation 22 (Peharz et al., 2016)) ▷ computation reused across sum units
    end for
  end for
end for
return $\{\boldsymbol{h}_{n,j} \ \ \forall n \in \{1, \dots, T\}, \forall j \in \{1, \dots, S\}\}$

On a separate note, when it comes to handling missing data, it is sufficient to marginalize out the missing evidence by substituting a 1 in place of the missing input units. This allows us to compute any marginal or conditional query (including the computation of the posterior probabilities) in the presence of partial evidence.

A.5 PROBABILISTIC SHORTCUT CONNECTIONS

In GSPN$_U$, we also propose probabilistic shortcut connections to set the shared parameters $\boldsymbol{\omega}^L$ as a convex combination of those at height $\{\boldsymbol{\omega}^0, \ldots, \boldsymbol{\omega}^{L-1}\}$. For instance, for a continuous *r.v.* $\tilde{\boldsymbol{X}}_n^L$ of the root node n, a modeling choice would be to take the mean of the $L-1$ Gaussians implementing the distributions, leveraging known statistical properties to obtain

$$P_{\boldsymbol{\omega}^L}(\tilde{\boldsymbol{X}}_n^L \mid Q_n^L = i) \stackrel{\text{def}}{=} \mathcal{N}\left(\cdot\,; \sum_{\ell=1}^{L-1} \frac{\mu_i^\ell}{L-1}, \sum_{\ell=1}^{L-1} \frac{(\sigma_i^\ell)^2}{(L-1)^2}\right), \;\; \boldsymbol{\omega}^\ell = (\mu_1^\ell, \sigma_1^\ell, \ldots, \mu_C^\ell, \sigma_C^\ell). \quad (9)$$

Instead, when the *r.v.* is categorical, we consider a newly parametrized Categorical distribution:

$$P_{\boldsymbol{\omega}^L}(\tilde{\boldsymbol{X}}_n^L \mid Q_n^L = i) \stackrel{\text{def}}{=} Cat\left(\cdot\,; \sum_{\ell=1}^{L-1} \frac{\boldsymbol{\omega}_i^\ell}{L-1}\right), \boldsymbol{\omega}_i^\ell \in C\text{-1-probability simplex} \;\; \forall i \in [1, C]. \quad (10)$$

Akin to residual connections in neural networks (Srivastava et al., 2015; He et al., 2016), these shortcut connections mitigate the vanishing gradient and the degradation (accuracy saturation) problem, that is, the problem of more layers leading to higher training error. In the experiments, we treat the choice of using residual connections as a hyper-parameter and postpone more complex design choices, such as *weighted* residual connections, to future work.

A.6 HYPER-PARAMETERS TRIED DURING MODEL SELECTION

The following tables report the hyper-parameters tried during model selection.

	Embedding Construction						
	C / latent dim	# layers	learning rate	batch size	# epochs	ES patience	avg emission params across layers
GAE	32, 128, 256	2,3,5	0,1, 0.01	1024	100	50	
DGI	32, 128, 256	2,3,5	0,1, 0.01	1024	100	50	
GSPN	5,10,20,40	5,10,20	0.1	1024	100	50	true, false

	Graph Predictor							
							global pooling	w. decay (MLP) dropout (GIN)
MLP	8,16,32,64	1	0.01	1024	1000	500	sum, mean	0, 0.0001
GIN	32,256,512	2,5	0.01, 0.0001	8,32,128	1000	500	sum, mean	0., 0.5

Table 4: Scarce supervision experiments. We found that too large batch sizes caused great instability in GIN's training, so we tried different, smaller options. DGI and GAE used the Atom Embedder of size 100 provided by OGBG, whereas GSPN deals with categorical attributes through a Categorical distribution. The range of hyper-parameters tried for GAE and DGI follows previous works.

	C	# layers	learning rate	batch size	# epochs	ES patience	avg emission params across layers
GAUSSIAN	-	-	-	-	-	-	-
GMM	5,15,20,40	1	0,1, 0.01	32	200	50	-
GSPN	5,15,20,40	2	0,1, 0.01	32	200	50	false

Table 5: Missing data experiments. Gaussian mixtures of distributions are initialized with k-means on the first batch of the training set. The maximum variance at initialization is set to 10.

	C / latent dim	# layers	learning rate	batch size	# epochs	ES patience	avg emission params across layers
			Embedding Construction				
GSPN$_U$	5,10,20	5, 10, 15, 20	0,1	32	500	50	true, false
			Graph Predictor				
							global pooling
MLP	8, 16, 32, 128	1	0.001	32	1000	200	sum, mean
GSPN$_S$	C=5,10,20 C_g=32,128	1	0.001	32	1000	200	sum, mean

Table 6: Graph classification experiments. The emission distribution is categorical for NCI1 and univariate Gaussian otherwise. Gaussian mixtures of distributions to be learned (social datasets) are initialized using k-means on the training set.

A.7 Datasets Statistics

Below we report the set of datasets used in our work together with their characteristics.

	# graphs	# vertices	# edges	# vertex attributes	task	metric
benzene	527984	12.00	64.94	1 categorical + 6 cont.	Regression(1)	MAE/NLL
ethanol	455093	9.00	36.00	3 (1 cat.) + 6 cont.	Regression(1)	MAE/NLL
naphthalene	226256	18.00	127.37	3 (1 cat.) + 6 cont.	Regression(1)	MAE/NLL
salicylic acid	220232	16.00	104.13	3 (1 cat.) + 6 cont.	Regression(1)	MAE/NLL
toluene	342791	15.00	96.15	3 (1 cat.) + 6 cont.	Regression(1)	MAE/NLL
malonaldehyde	893238	9.00	36.00	3 (1 cat.) + 6 cont.	Regression(1)	MAE/NLL
uracil	133770	12.00	64.44	4 (1 cat.) + 6 cont.	Regression(1)	MAE/NLL
ogbg-molpcba	437929	26.0	28.1	83 (9 cat.)	Multi-Label Classification(128)	AP
NCI1	4110	29.87	32.30	37 (1 cat.)	Classification(2)	ACC
REDDIT-B	2000	429.63	497.75	1 cont.	Classification(2)	ACC
REDDIT-5K	5000	508.52	594.87	1 cont.	Classification(5)	ACC
COLLAB	5000	74.49	2457.78	1 cont.	Classification(3)	ACC

Table 7: Dataset statistics.

A.8 Graph Classification Experiments: Ablation and Hyper-parameter Studies

Ablation Study The following table shows the performance of GSPN$_{U+DS}$ when removing the use shortcut connections from the hyper-parameter space. The results show that using shortcut connections consistently leads to better mean classification accuracy on these tasks.

	NCI1	**REDDIT-B**	**REDDIT-5K**	**COLLAB**
GSPN$_{U+DS}$ (no shortcut)	76.5 ± 1.9	88.9 ± 3.9	52.3 ± 5.2	75.7 ± 2.6
GSPN$_{U+DS}$	76.6 ± 1.9	90.5 ± 1.1	55.3 ± 2.0	78.1 ± 2.5

Table 8: Impact of shortcut connections on the creation of unsupervised embeddings for graph classification.

Impact of Hyper-parameters Figure 4 shows how the validation performance of GSPN$_S$ changes for specific hyper-parameters C and C_G as we add more layers. Please refer to the main text for a discussion.

A.9 Scarce Supervision Experiments: Additional Visualizations

Figure 5 provides a few, randomly picked examples of molecules from the *ogbg-molpcba* dataset, modified to show the relative change in pseudo log-likelihood of some of the vertices according to GSPN.

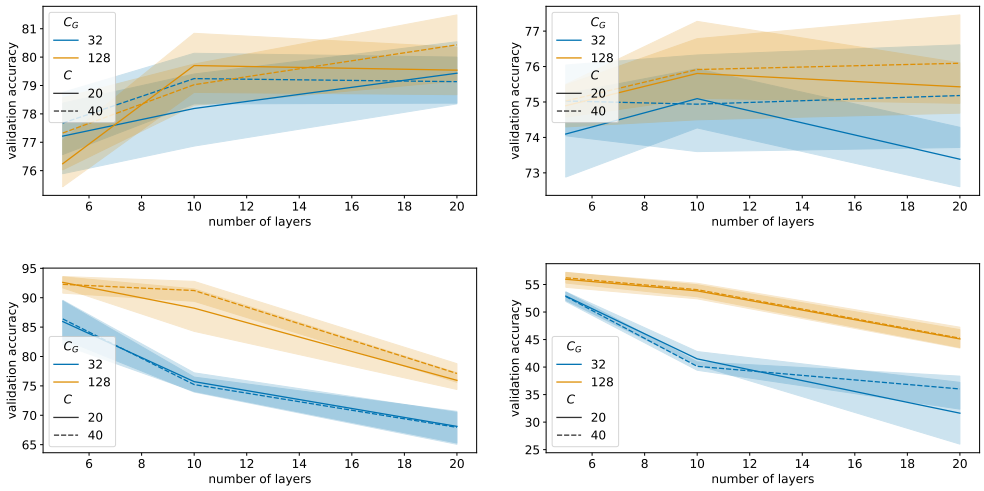

Figure 4: Impact of GSPN$_S$ layers, C and C_G on NCI1 (top left), COLLAB (top right), REDDIT-BINARY (bottom left), and REDDIT-5K (bottom right) performances, averaged across all configurations in the 10 outer folds.

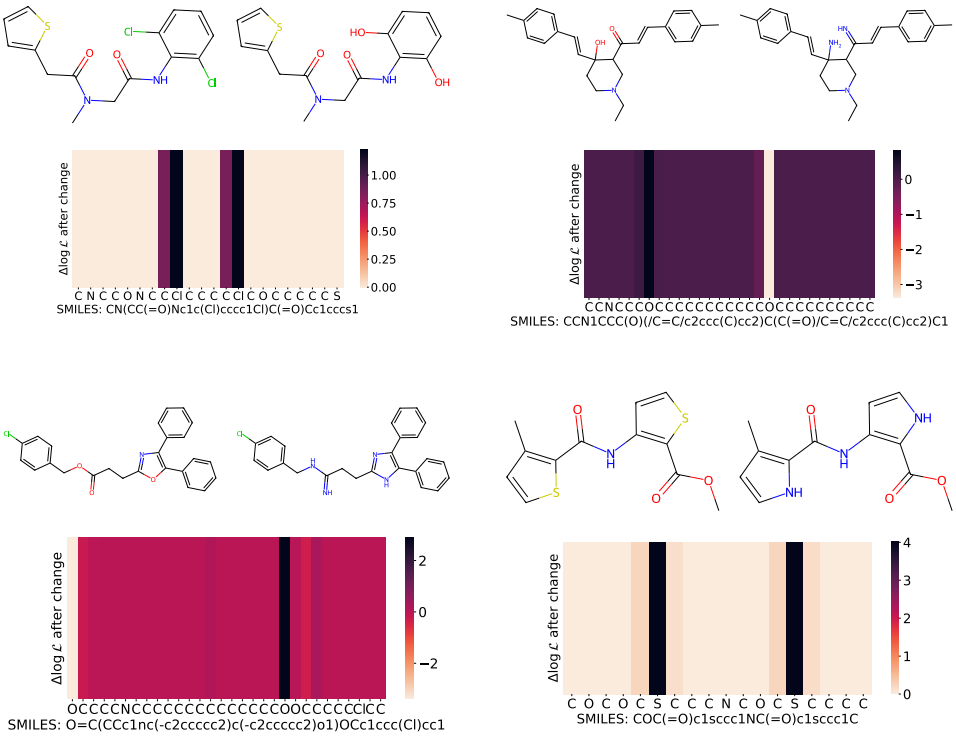

Figure 5: Additional visualizations of randomly picked molecules from *ogbg-molpcba* for a specific GSPN configuration. The heatmap shows pseudo log-likelihood variations in the vertices when operating specific atomic changes (from left to right). Nearby atoms are affected by the change as well, and this allows us to understand which groups are more common, or make more sense, than others.

A.10 TIME COMPARISON

Table 9 shows the time comparison of forward and backward passes on a batch of size 32 between GSPN and GIN. We used 10 layers for both architectures and adapted the hidden dimensions to obtain a comparable number of parameters. Despite the GIN's implementation being more sample efficient than GSPN, the table confirms our claims on the asymptotic complexity of GSPN.

	# parameters		forward time (ms)		backward time (ms)	
	GSPN	GIN	GSPN	GIN	GSPN	GIN
NCI1	36876	36765	30	12	17	14
REDDIT-B	34940	35289	35	14	24	15
REDDIT-5K	34940	35289	35	14	22	15
COLLAB	34940	35289	36	14	21	15

Table 9: Time comparison between GSPN and GIN on graph classification tasks.