A Unified Framework for Human-Allied Learning of Probabilistic Circuits

Athresh Karanam*, Saurabh Mathur*, Sahil Sidheekh*, Sriraam Natarajan

The University of Texas at Dallas
{Athresh.Karanam, SaurabhSanjay.Mathur, Sahil.Sidheekh, Sriraam.Natarajan}@utdallas.edu

Abstract

Probabilistic Circuits (PCs) have emerged as an efficient framework for representing and learning complex probability distributions. Nevertheless, the existing body of research on PCs predominantly concentrates on data-driven parameter learning, often neglecting the potential of knowledge-intensive learning, a particular issue in data-scarce/knowledge-rich domains such as healthcare. To bridge this gap, we propose a novel unified framework that can systematically integrate diverse domain knowledge into the parameter learning process of PCs. Experiments on several benchmarks as well as real-world datasets show that our proposed framework can both effectively and efficiently leverage domain knowledge to achieve superior performance compared to purely data-driven learning approaches.

Code — https://github.com/athresh/unified-constraints-pc/

Introduction

Probabilistic Circuits (PCs) have emerged as powerful tools for representing complex probability distributions. Their key advantage is enabling efficient and exact probabilistic inference (Choi, Vergari, and Van den Broeck 2020). Recent work has focused on improving PC expressivity, and allowing them to compete with deep generative models (DGMs) (Peharz et al. 2020a; Liu, Zhang, and den Broeck 2023; Correia et al. 2023; Sidheekh, Kersting, and Natarajan 2023). However, similar to DGMs, this data-driven approach often leads to models that are highly reliant on large datasets, make strong assumptions, and are susceptible to outliers (Ventola et al. 2023). Purely data-driven PCs often struggle in data-scarce and noisy domains, exhibiting overfitting and reduced generalizability. While traditional techniques like regularization can address overfitting to some extent, they often do so by introducing strong and sometimes arbitrary assumptions about the data distribution (Domingos 1999). This can limit the model's ability to capture the true underlying relationships between variables.

Incorporating domain knowledge and expert insights into PC learning offers a compelling solution to these problems. Domain experts can provide invaluable insights that go beyond the raw data, grounded in experience and contextual understanding. This knowledge can not only mitigate the risk of overfitting without sacrificing expressive power but also tailor models to specific domain constraints. For instance, fairness and privileged information might be crucial factors in real-world decision-making, requiring models to incorporate such constraints. Knowledge-intensive learning strategies have demonstrably enhanced both discriminative and generative models in various contexts (Kokel et al. 2020; Odom et al. 2015; Altendorf, Restificar, and Dietterich 2005; de Campos, Tong, and Ji 2008; Yang and Natarajan 2013; Mathur, Gogate, and Natarajan 2023). These models perform more reliably in scenarios with limited or suboptimal data. However, the concept of integrating human expertise and domain knowledge remains *relatively unexplored in the context of the general class of PCs*.

This work addresses this problem by introducing a framework for incorporating domain knowledge into PC learning. We first develop a unified framework that allows encoding *various types of knowledge* as probabilistic domain constraints. A key aspect of our work is that we can model any constraint **as long as its violation can be measured by a differentiable function**. We then demonstrate how **some** of these constraints, including equality constraints for generalization and privileged information, and inequality constraints for handling class imbalance, monotonicity, and variable interactions, can be seamlessly integrated into PC learning. Our method can be interpreted as **frustratingly easy** as with Shi et al. (2022), however, it works well and is generalizable, precisely why simpler learning methods should be favored over unnecessarily complex ones (Rudin et al. 2024; Rudin 2019). This approach makes PCs more adaptable to real-world scenarios with limited data and specific modeling requirements. The key tenets of our framework are (1) Expressivity - allowing incorporation of a wide variety of well-known domain knowledge, (2) Effectiveness - consistently improving model performance by leveraging the domain knowledge and (3) Simplicity - allowing domain experts to easily quantify their domain knowledge through equality and inequality constraints.

We make the following key contributions:(1) We propose a unified framework that subsumes several types of domain knowledge and allows encoding them as domain constraints

*These authors contributed equally.

and demonstrate six particular instantiations of these constraints. (2) We propose an augmented objective function that effectively incorporates these domain constraints and can be optimized through generic parameter learning algorithms for PCs. (3) We experimentally validate the added utility of our approach on several benchmark and real-world scenarios where data is limited but knowledge is abundant.

Background

Notations: We use X to denote a random variable and x to denote an assignment of value to X. Sets of random variables are denoted as $\mathbf{X}$ and their values as $\mathbf{x}$. We denote the subset of variables in set k as $\mathbf{X}_k$ and those not in k as $\mathbf{X}_{-k}$. We use $\mathcal{M} = (\theta, G)$ to denote a probabilistic circuit having structure G and parameterized by θ. We use $P(\mathbf{X})$ to denote the joint probability distribution over $\mathbf{X}$.

Probabilistic Circuits (PCs) (Choi, Vergari, and Van den Broeck 2020) are generative models that represent probability distributions in the form of computational graphs comprising three types of nodes - sums, products, and leaves. Each node in the graph represents a (possibly unnormalized) distribution over a set of variables, known as its scope. The internal nodes in the graph are sums and products. Sum nodes compute a convex sum of the distributions modeled by its children, representing a mixture distribution. Product nodes compute a product over the outputs of their children, representing a factorized distribution over their scopes. Leaf nodes encode simple tractable distributions such as a Gaussian. A PC is evaluated bottom up and the output of its root node gives the modeled joint probability density.

The structure of a PC must satisfy certain properties for it to be able to perform exact inference tractably, two of which are *Smoothness* and *Decomposability*. Smoothness requires that the scope of each child of a sum node be identical. *Decomposability* requires that the scopes of the children of a product node be disjoint. Formally, a PC $\mathcal{M}$ is a tuple $\langle G, \theta \rangle$ where G is a DAG consisting of sum, product, and leaf nodes. The distributions over a sum node s having scope S_s and product node p with scope S_p are,

$$P_s(\mathbf{X}_{S_s} = \mathbf{x}_{S_s}) = \sum_{j \in \text{Ch}(s)} \theta_{sj} P_j(\mathbf{X}_{S_j} = \mathbf{x}_{S_j}),$$

$$P_p(\mathbf{X}_{S_p} = \mathbf{x}_{S_p}) = \prod_{j \in \text{Ch}(p)} P_j(\mathbf{X}_{S_j} = \mathbf{x}_{S_j})$$

where $\text{Ch}(p)$ denotes the children of node p. The distribution at a leaf l having scope S_l is assumed to be tractable and parameterized by θ_l. Simple distributions such as Bernoulli and Gaussian are commonly used as leaf distributions.

The structure and parameters of PCs can be jointly learned from data. Structure learning algorithms typically use heuristics to recursively learn the PC (Gens and Domingos 2013; Rooshenas and Lowd 2014; Dang, Vergari, and Van den Broeck 2020; Adel, Balduzzi, and Ghodsi 2015; Peharz, Geiger, and Pernkopf 2013). Alternatively randomized structures (Mauro et al. 2017; Peharz et al. 2020b,a) that can be easily overparameterized and scaled using GPUs are used, shifting the focus to parameter learning using data-driven techniques, resulting in deep and expressive PCs (Correia et al. 2023; Sidheekh, Kersting, and Natarajan 2023; Liu, Zhang, and den Broeck 2023; Liu et al. 2023).A detailed review of such parameterizations can be found in (Sidheekh and Natarajan 2024). However, these advanced PCs often require large amounts of data to learn effectively, limiting their use in scarce and noisy data settings.

Knowledge-based Learning. The role of domain knowledge as constraints becomes crucial in guiding probabilistic model construction in several domains. These constraints concisely encode information about general trends in the domain and serve as effective inductive biases, yielding more useful and more accurate probabilistic models, especially in noisy and sparse domains (Towell and Shavlik 1994; van der Gaag, Bodlaender, and Feelders 2004; Altendorf, Restificar, and Dietterich 2005; Yang and Natarajan 2013; Odom et al. 2015; Kokel et al. 2020; Plajner and Vomlel 2020).

Prior works have explored incorporating such constraints in PCs in a hard or exact way. Semantic Probabilistic Layers (Ahmed et al. 2022) guarantee exact symbolic constraint satisfaction for structured-output prediction but are limited to propositional logic and hard constraints. Ghandi, Quost, and de Campos (2024) adjust the leaves of trained PCs for probabilistic constraint satisfaction but do not consider parameter adaptation during training. In contrast, we aim to learn more accurate PCs from data by exploiting diverse forms of domain knowledge to enhance generalization in noisy or sparse data settings.

While prior works have explored the use of domain knowledge to learn more accurate PCs, they have either been limited to equality constraints (Papantonis and Belle 2021) or specific sub-classes of PCs (Galindez Olascoaga et al. 2020; Mathur, Gogate, and Natarajan 2023). As far as we are aware, diverse forms of domain knowledge have *not been used previously to learn the parameters of the general class of PCs*. To bridge this gap, we propose a unified framework for representing and learning from diverse forms of domain knowledge including generalization constraints, qualitative influence statements (Altendorf, Restificar, and Dietterich 2005), context-specific independence relations (Boutilier et al. 1996), class imbalance tradeoffs (Yang et al. 2014), and privileged information (Pasunuri et al. 2016).

Learning PCs with Domain Knowledge

We aim to solve the following problem:

Given: Dataset $\mathcal{D}$ over random variables $\mathbf{X}$, and a specification of domain knowledge $\mathcal{K}$.
To Do: Learn a probabilistic circuit $\mathcal{M}$ that accurately models $P(\mathbf{X})$.

Mathematically, the above problem can be expressed as the following constrained optimization:

$$\arg\max_{\mathcal{M}} \mathcal{L}(\mathcal{M}, \mathcal{D}) \text{ s.t. } \mathcal{M} \text{ does not violate } \mathcal{K} \quad (1)$$

where $\mathcal{L}(\mathcal{M}, \mathcal{D})$ is the log-likelihood of $\mathcal{D}$ with respect to $\mathcal{D}$ and is given by the following equation

$$\mathcal{L}(\mathcal{M}, \mathcal{D}) = \sum_{\mathbf{x} \in \mathcal{D}} \log P_{\mathcal{M}}(\mathbf{X} = \mathbf{x}) \tag{2}$$

This formulation reduces to standard maximum likelihood estimation when there are no domain constraints ($\mathcal{K} = \emptyset$). Similarly, including a simple constraint on model complexity ($\mathcal{K}$ ="$\mathcal{M}$ is not too complex.") recovers maximum likelihood estimation with a basic regularization term. We first present an encoding scheme to encode commonly used forms of knowledge as constraints on the PC and then we present an algorithm that learns a PC using these constraints.

Encoding knowledge as constraints

We now demonstrate that commonly used forms of knowledge can be represented as either equality or inequality constraints on marginal and conditional queries on the PC. Specifically, we consider generalization, monotonicity, context-specific independence, class imbalance, synergy, and privileged information. We formally define equality and inequality constraints as follows:

Definition 1. An equality constraint $\langle f, g, \mathcal{S} \rangle$ states that $f(\mathbf{x}) = g(\mathbf{x}') \forall \mathbf{x}, \mathbf{x}' \in \mathcal{S}$, where $f(\mathbf{x})$ and $g(\mathbf{x})$ are tractable, differentiable functions of the PC and $\mathcal{S} \subseteq \mathbf{X}^2$ is a set of pairs of data points.

Definition 2. An inequality constraint $\langle f, g, \mathcal{S} \rangle$ states that $f(\mathbf{x}) > g(\mathbf{x}') \forall \mathbf{x}, \mathbf{x}' \in \mathcal{S}$, where $f(\mathbf{x})$ and $g(\mathbf{x})$ are tractable, differentiable functions of the PC and $\mathcal{S} \subseteq \mathbf{X}^2$ is a set of pairs of data points.

We use the case of learning a PC to model the risk of Gestational Diabetes (GD) using data from a multi-center clinical study as a running example. Ideally, such a PC should capture the relationships between various risk factors such as age, race, BMI, family history, genetic predisposition, and exercise level, by modeling their joint distribution. The following forms of domain knowledge can be used for learning:

1. **Generalization constraints (GC).** These constraints capture inherent symmetries in the data distribution, such as exchangeability (Lüdtke, Bartelt, and Stuckenschmidt 2022) and permutation invariance (Zaheer et al. 2017). For instance, subjects from the same center might have similar distributions due to study design. We can express this as:

$$P(\mathbf{x}) = P(\mathbf{x}'), \quad \forall (\mathbf{x}, \mathbf{x}') \in \mathcal{D}^2 \text{ s.t. } sim(\mathbf{x}, \mathbf{x}')$$

where $sim(\mathbf{x}, \mathbf{x}')$ indicates if $\mathbf{x}$ and $\mathbf{x}'$ are similar based on predefined criteria (e.g., belonging to the same center).

2. **Context-Specific Independence relations (CSI).** These relations are a generalization of conditional independence relationships between variables. For example, BMI might be independent of age only for patients with GD. We can represent this as:

$$P(x_i \mid \mathbf{x}_k) = P(x_i \mid x_j, \mathbf{x}_k), \quad \forall \mathbf{x} \in \text{Dom}(\mathbf{X}) \text{ s.t. } \mathbf{x}_k = c$$

where $\mathbf{x}_k = c$ denotes a specific value assignment to variables in $\mathbf{X}_k$, defining the context (e.g., having GD) where independence between X_i and X_j holds.

3. **Preference constraints (PF)**. Medical experts might provide a set of probabilistic logical conditions indicative of high GD risk. We denote this set as $R = \{(r_1, p_1), \ldots, (r_M, p_M)\}$, where each pair (r, p) consists of a logical condition r over variables $\mathbf{X}$ and its associated probability p. These constraints can be expressed as

$$P(x_i = 1 \mid \mathbf{x}_{-i}) = p \quad \forall \mathbf{x} \models r, \ \forall (r, p) \in R$$

4. **Class-imbalance tradeoff (CT)** In GD screening, minimizing false negatives is crucial since patients predicted as low risk might not undergo further clinical testing. We can express such false negative constraints as

$$t_0 > P(X_i = 0 \mid \mathbf{x}_{-i}), \quad \forall \mathbf{x} \in \mathcal{D} \text{ s.t. } x_i = 1$$

where t_0 is the threshold such that $P(X_i = 0 \mid x_{-i}) > t_0$ is predicted as negative. These constraints ensure that the model is cautious in predicting a negative outcome, thus reducing the risk of false negatives. Constraints to minimize false positives can be formulated similarly.

5. **Monotonic Influence Statements (MISs)** The statements express positive or negative monotonic relationships between pairs of variables. For example, the risk of GD might increase with an increase in age. We can represent positive monotonic influence of X_j on X_i ($X_j \prec^{M+} X_i$) as:

$$P(X_i \leq x_i \mid x_j) > P(X_i \leq x_i' \mid x_j') \\ \forall \mathbf{x}, \mathbf{x}' \text{ s.t. } x_i = x_i', \ x_j' > x_j$$

(6) Synergistic influence statements can be encoded similar to monotonicities while (7) Privileged information can be encoded similar to CSIs. We defer the details about both of these constraints to the supplementary material.

Defining the penalty function

Having demonstrated that **seven commonly used forms of knowledge** can be represented as equality or inequality constraints on marginal and conditional queries on the PC, we now present a learning algorithm that uses these constraints to learn PCs. To formalize the constrained optimization problem, we define the notion of a *penalty function* that quantifies the severity of domain knowledge violation in the distribution induced by the PC.

Definition 3. A function $\zeta : M \mapsto \mathbb{R}_0^+$ is a penalty function corresponding to domain knowledge $\mathcal{K}$ if it maps a PC $\mathcal{M} \in M$ to a non-negative real number $\zeta(\mathcal{M})$ that quantifies the extent of the violation of knowledge $\mathcal{K}$ such that

1. $\zeta(\mathcal{M}) = 0 \iff \mathcal{M}$ does not violate $\mathcal{K}$.
2. $\zeta(\mathcal{M}) < \zeta(\mathcal{M}') \iff \mathcal{M}$ violates $\mathcal{K}$ to a lesser extent than $\mathcal{M}'$.
3. The total violation in a model $\mathcal{M}$ due to two penalty functions ζ and ζ' is $\zeta(\mathcal{M}) + \zeta'(\mathcal{M})$

Using this notation, we can rewrite the optimization problem in equation (1) as

$$\arg\max_{\mathcal{M}} \mathcal{L}(\mathcal{M}, \mathcal{D}) \text{ s.t. } \zeta(\mathcal{M}) = 0 \tag{3}$$

where ζ measures the model's deviation from domain knowledge $\mathcal{K}$. For computational reasons, we focus on

Algorithm 1: Learn PC Parameters with Knowledge

input : Structure of PC G, Data $\mathcal{D}$, Penalty violation function $\zeta : M \mapsto \mathbb{R}_0^+$, Maximum number of tries $t_{\max}$, Penalty weight control γ,
output: Parameters for PC θ

initialize: $\theta = \arg\max_{\theta} \mathcal{L}(\theta; G, \mathcal{D}), t = 1$ ▷ start with maximum likelihood solution
$\lambda_1 = 1$
while $\zeta(G, \theta) \neq 0$ *and* $t \leq t_{max}$ **do**
  ▷ while constraints are not satisfied
  $\theta = \arg\max_{\theta} \mathcal{L}(\theta; G, \mathcal{D}) - \lambda_t \zeta(G, \theta)$
  $\lambda_{t+1} = \lambda_t \times \gamma$ ▷ increase penalty weight
  $t = t + 1$
end
return θ

learning the model parameters for a PC with a given structure ($\mathcal{G}$). The optimization problem becomes finding PC parameters (θ) that maximize the likelihood while adhering to the domain knowledge:

$$\arg\max_{\theta} \mathcal{L}(\langle \mathcal{G}, \theta \rangle, \mathcal{D}) \text{ s.t. } \zeta(\langle \mathcal{G}, \theta \rangle) = 0 \tag{4}$$

However, since domain knowledge elicited from experts might not be perfectly consistent or fully compatible with the given PC structure, equation (3) might not have any feasible solution. We address this by using the penalty method to find a solution closest to the feasible region (Luenberger and Ye 2016; Mathur, Gogate, and Natarajan 2023). Algorithm 1 presents the parameter learning procedure, which solves a series of optimizations of the form

$$\theta_t^* = \arg\max_{\theta} \mathcal{L}(\langle \mathcal{G}, \theta \rangle, \mathcal{D}) - \lambda_t \zeta(\langle \mathcal{G}, \theta \rangle) \tag{5}$$

where θ_t^* is the solution of the tth optimization and λ_t is the penalty weight in the tth optimization. The initial value of the penalty weight, λ_0 is set to 0 (corresponding to the maximum likelihood solution), and the value of λ_t for each subsequent optimization problem is increased using the penalty control parameter γ and is set to γ^{t-1}. Each optimization where $t > 0$ is initialized with the solution from the previous problem θ_{t-1}. Note that while the objective function appears simple (as a function of likelihood with penalty), this has been widely used for structure learning in graphical models (Schwarz 1978), and more recently, even in domain adaptation. Our empirical evaluation is yet another proof that one does not necessarily need complex objective functions if the simpler ones are both effective and by definition, efficient.

Theorem 1. *If the PC $\mathcal{M}$ is smooth, decomposable, and deterministic, and ζ is a differentiable, concave function of θ, then algorithm 1 is guaranteed to converge to the optimal feasible solution of equation* (4)*, if one exists, as $t_{max} \to \infty$.*

Proof. The likelihood of deterministic PCs is a concave function of the parameters (Choi, Vergari, and Van den Broeck 2020). The penalty method is guaranteed to converge to an optimal solution of the constrained optimization problem if one exists, the objective function is concave and the penalty function is differentiable (Luenberger and Ye 2016). □

We define the penalty function for equality constraints as $\zeta(G, \theta) = \sum_{(\mathbf{x}, \mathbf{x}') \in \mathcal{S}} |\delta(\mathbf{x}, \mathbf{x}')|$ and for inequality constraints as $\zeta(G, \theta) = \sum_{(\mathbf{x}, \mathbf{x}') \in \mathcal{S}} \max\{0, \delta(\mathbf{x}, \mathbf{x}') + \epsilon\}^2$. Here, $\delta(\mathbf{x}, \mathbf{x}') = g(\mathbf{x}) - f(\mathbf{x})$ and $\epsilon > 0$ is a margin parameter used to control the minimum extent of the inequality. In practice, we might approximate the exact constraint using a subset of $\mathcal{S}' \subseteq \mathcal{S}$ of size γ_{size}. In the case of multiple pieces of knowledge, we solve the constrained optimization stage-wise. We add constraints corresponding to each piece of knowledge one by one, using the solution at each stage as initialization for the next. We use algorithm 1 to find the solution to the sub-problem at each stage by defining the penalty function ζ as the sum of the penalty functions corresponding to the current set of constraints.

Experimental Evaluation

Our framework is generic and agnostic to the type of PC used. Thus, to empirically validate the effectiveness of incorporating domain constraints we consider two different instantiations of deep PCs - (i) *RatSPN* (Peharz et al. 2020b) and (ii) *EinsumNet* (Peharz et al. 2020a). We implement both models with and without domain constraints for a comparative analysis. We design experiments over various datasets and scenarios to answer the following questions empirically:

(**Q1**) Can domain knowledge be incorporated faithfully into the parameter learning of PCs?

(**Q2**) Does incorporating knowledge improve the generalization performance of PCs?

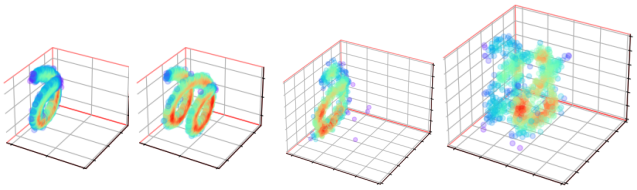
(**Q3**) Can the proposed framework exploit domain knowledge in heterogenous forms to learn more accurate PCs?

(**Q4**) How sensitive is the approach to the hyperparameters governing the size of domain sets and penalty weight? Further, is it robust to noisy or redundant advice?

(**Q5**) Does the proposed method learn more accurate models on real-world data?

Experimental Setup. To answer these questions, we compare our knowledge-intensive learning method to purely data-driven approaches. We defer further implementation details to the supplementary material. We used three types of data sets for our experiments: synthetic, benchmark, and real-world clinical data.

Synthetic data sets. We used two types of synthetic data sets: four Bayesian Network (BN)-based datasets and a manifold-based data set. We derived data sets from 4 BNs: Earthquake (Korb and Nicholson 2010), Asia (Lauritzen 1988), Sachs (Sachs et al. 2005), and Survey (Scutari and Denis 2021). For each BN, we constructed a data set by sampling 100 data points. To simulate perfect domain knowledge, we then extracted two conditional independence relations from each BN. These relations were then translated into CSI constraints.



(a) Train (b) Test (c) *EinsumNet* (d) *EinsumNet+GC*

Figure 1: **3D Helix dataset**: Visualization of the 3D Helix dataset. The train dataset (a) consists of a single helix of length 2π with Gaussian noise added to it. The test dataset (b) consists of a helix of length 4π with Gaussian noise added to it. (c) and (d) visualizes 1000 samples generated from *EinsumNet* with and without incorporating the Generalization Constraint (*GC*), respectively.

		RatSPN	*RatSPN+Constraint*
BN	asia	-483.3 ± 4.1	$\mathbf{-313.2 \pm 3.9}$
	sachs	-1097.5 ± 8.8	$\mathbf{-861.2 \pm 8.7}$
	survey	-611.7 ± 7.2	$\mathbf{-476.6 \pm 6.6}$
	earthquake	-272.0 ± 2.4	$\mathbf{-121.8 \pm 2.1}$
UCI	breast-cancer	-2110.8 ± 15.6	$\mathbf{-1271.5 \pm 14.6}$
	diabetes	-7010.3 ± 31.0	$\mathbf{-5070.3 \pm 481.8}$
	thyroid	-351.5 ± 6.1	$\mathbf{-200.5 \pm 23.2}$
	heart-disease	-931.7 ± 15.0	$\mathbf{-739.8 \pm 7.2}$
RW	numom2b-a	-14573.9 ± 69.9	$\mathbf{-7288.2 \pm 1.6}$

Table 1: **Quantitative Evaluation:** Mean test log-likelihood of *RatSPN* trained with and without constraints on the Bayesian Network (BN), UCI Benchmark and Real-World (RW) datasets, $\pm$ standard deviation across 3 independent trials.

We used a 3D Helix Manifold to construct the second type of synthetic data set (Figure 1) (Sidheekh et al. 2022). This dataset presents difficulties due to its intricate spiral structure. However, its inherent symmetry can be exploited to define a generalization constraint (GC). We encoded the helical symmetry by specifying that points separated by 2π along the x-axis and lying on a helix manifold of unit radius have similar density. We use 100 pairs of datapoints of the form $((x, sin(x), cos(x)), (x + 2\pi, sin(x + 2\pi), cos(x + 2\pi)))$.

Benchmark data sets. We used two kinds of benchmark data sets – four UCI benchmark data sets and two point cloud-based data sets. We used 4 data sets from the UCI Machine Learning repository, namely, breast-cancer, diabetes, thyroid, and heart-disease. We used the monotonic influence statements that were used in prior work by Yang and Natarajan (2013) as domain knowledge for these data sets.

We also used the MNIST (Deng 2012) and fashion-MNIST image datasets to construct point cloud representations following (Zhang, Hare, and Prugel-Bennett 2019). We transformed each image into a set-based representation by sampling the locations of a small set of foreground pixels. The resulting 2D point cloud representation inherently exhibits permutation invariance – a symmetry difficult to model even for advanced generative models (Li et al. 2019; Kim et al. 2021) such as Generative Adversarial Networks and Variational Autoencoders. We refer to these datasets as full set data (e.g., Set-MNIST-Full). Additionally, to simulate a data-scarce scenario, we further divided the MNIST dataset into two subsets: (i) Set-MNIST-Even comprising only even digits and (ii) Set-MNIST-Odd comprising only odd digits. We encoded permutation invariance as a GC by specifying the domain set criteria as $sim(\mathbf{x}, \mathbf{x}') = \mathbb{I}[\mathbf{x}' = \pi(\mathbf{x})]$, where $\pi(\mathbf{x})$ represents a permutation of $\mathbf{x}$. In practice, we defined the GC domain set by taking $\gamma_{size} = 2$ permutations for each sample in the dataset.

Real-World Clinical Data. To evaluate the performance of our framework on real-world data, we used data from the Nulliparous Pregnancy Outcomes Study: Monitoring Mothers-to-Be (nuMoM2b, (Haas, Parker et al. 2015)). We considered two subsets of the data, numom2b-a focusing on a single Adverse Pregnancy Outcome (APO) namely Gestational diabetes (GD), and numom2b-b focusing on three different APOs namely New Hypertension (NewHTN), Pre-eclampsia (PreEc), and Pre-term Birth (PTB).
The first data set consists of 3,657 subjects of white, European ancestry that have data for GD and its 7 risk factors: Age, BMI at the start of pregnancy, presence of Polycystic Ovary Syndrome (PCOS), physical activity (METs), presence of High Blood Pressure (HiBP), family history of diabetes (Hist), and a polygenic risk score (PRS) indicating genetic predisposition to diabetes. All risk factors except physical activity (METs) are known to positively monotonically influence the risk of GDM, while physical activity is known to have a negative monotonic influence. The second data set consists of 9,368 subjects of diverse racial and ethnic backgrounds that have data for three APOs: NewHTN, PreEc, PTB, and four risk factors: BMI, Age, Hist, and HiBP.

Results

(Q1) Faithful Incorporation of Domain Constraints.

Table 1 (rows 1-4) presents the test log-likelihood scores for RatSPN models trained on the BN-based data sets with and without domain knowledge encoded as CSIs. Notably, the degree of constraint violation for RatSPN models incorporating domain constraints remained consistently below 0.0001 across all datasets. This confirms our framework's ability to faithfully integrate valid domain knowledge. Moreover, RatSPN models trained with domain constraints performed better than those trained solely on data.

(Q2) Improvement in Generalization Performance.

We studied the effect on generalization performance due to two kinds of domain knowledge – generalization constraints (GC) and monotonic influence statements. We used the 3D Helix Manifold dataset and the point cloud data sets to evaluate the effect of GC. Table 2 shows the mean test log-likelihoods of the PCs trained on the 3D Helix Manifold data with and without domain knowledge encoded as GC. The baselines, EinsumNet and RatSPN, simply append the GC data points to the training data. In contrast, our proposed approach (EinsumNet+GC and RatSPN+GC) treats the GC as a constraint during training. We observed significant performance improvements for both models when incorporating the GC. This suggests that the GC enables the models to exploit the inherent symmetries within the data, allowing

	Helix	Set-MNIST-Even	Set-MNIST-Odd	Set-MNIST-Full	Set-Fashion-MNIST
RatSPN	-7.1 ± 1.8	-657.7 ± 15.9	-682.3 ± 16.7	-599.3 ± 3.2	-1495.0 ± 3.5
RatSPN+GC	$\mathbf{-3.6 \pm 0.1}$	$\mathbf{-562.2 \pm 0.4}$	$\mathbf{-555.1 \pm 0.4}$	$\mathbf{-566.1 \pm 0.1}$	$\mathbf{-1236.9 \pm 0.4}$
EinsumNet	-5.1 ± 0.2	-666.2 ± 13.7	-683.7 ± 19.9	-600.8 ± 4.8	-1413.9 ± 20.8
EinsumNet+GC	$\mathbf{-2.7 \pm 0.1}$	$\mathbf{-561.9 \pm 0.2}$	$\mathbf{-555.1 \pm 0.1}$	$\mathbf{-566.7 \pm 0.3}$	$\mathbf{-1235.8 \pm 0.6}$

Table 2: **Quantitative Evaluation**: Mean test log-likelihood of *RatSPN* and *EinsumNet* trained with and without the Generalization Constraint (*GC*) on the Helix and Set-MNIST datasets, $\pm$ standard deviation across 3 independent trials.

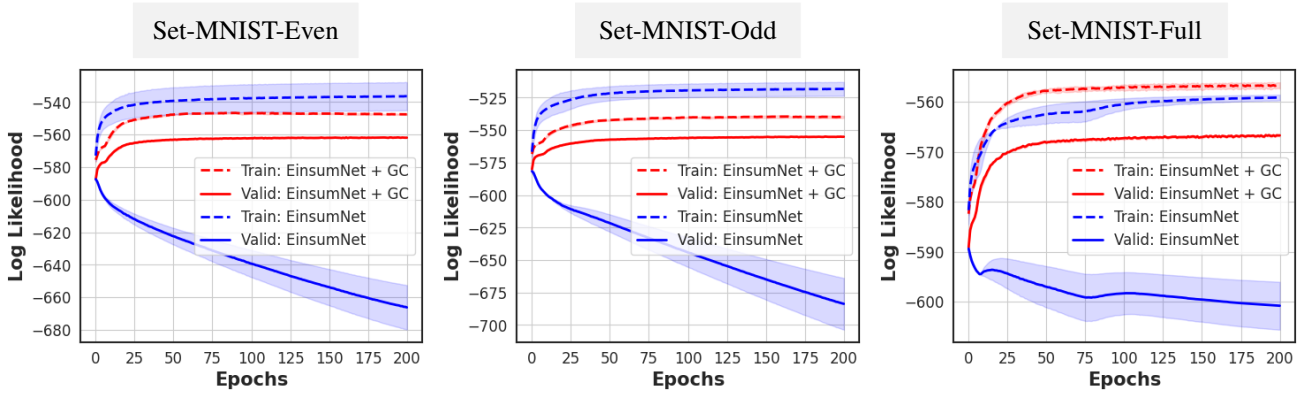


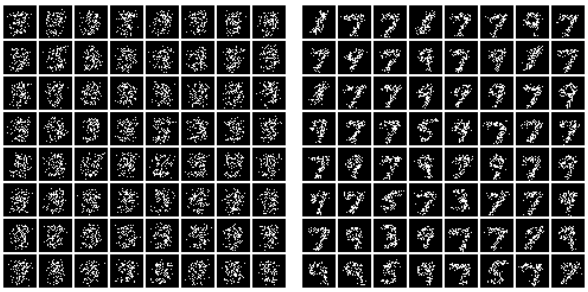
Figure 2: **Learning Curves:** Mean train and validation log-likelihoods of *EinsumNet* trained with (in red) and without (in blue) incorporating Generalization Constraint (*GC*) on the three Set-MNIST datasets, across training epochs. The shaded regions denote the standard deviation across 3 independent trials.

	RatSPN	+CSI	+CSI+MIS
earthquake	-272.0 ± 2.4	-137.7 ± 4.7	$\mathbf{-106.1 \pm 1.1}$
survey	-611.7 ± 7.2	-523.5 ± 4.3	$\mathbf{-470.9 \pm 6.6}$
asia	-483.3 ± 4.1	-320.5 ± 9.9	$\mathbf{-284.7 \pm 6.4}$
numom2b-b	-18281.2 ± 218.8	-15122.9 ± 201.7	$\mathbf{-14758.1 \pm 60.3}$

Table 3: The test log-likelihoods for the RatSPN learned from data, with one form of knowledge and with two forms of knowledge averaged over 3 independent trials

them to generalize to unseen regions with similar structure. Notably, this improvement required only a small number of samples from the constraint's domain set. Additionally, including these data points directly in the training data without structured GC integration did not lead to better generalization (Table 2). Figure 1 visually confirms this, as samples generated by EinsumNet+GC more closely resemble the test data distribution compared to those from EinsumNet.

Similar results were observed for set-based image datasets (MNIST and fashion-MNIST) (Table 2). Incorporating GC significantly improved performance for both models. This is further supported by the quality of samples generated with and without GC (Figure 3). The impact of GC on model training is also evident in the learning curves for Set-MNIST datasets (Figure 2). In the absence of GC (particularly for Set-MNIST-Even and Set-MNIST-Odd with fewer data points), the model overfits as shown by the increasing training log-likelihood and decreasing validation log-likelihood. Conversely, incorporating GC enables the model to leverage symmetries and achieve better generalization. Similar visualizations for other datasets are provided in the supplementary material.

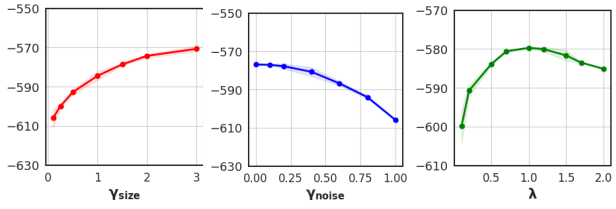


(a) *EinsumNet* (b) *EinsumNet+GC*

Figure 3: **Qualitative Evaluation**: Visualization of randomly sampled datapoints generated by *EinsumNet* trained with and without the Generalization Constraint (*GC*) on Set-MNIST-Odd.

To evaluate the effect of monotonic influence statements, we used four UCI benchmark datasets. Table 1 shows the improvement in the test log-likelihood of RatSPN trained on UCI data sets when incorporating domain constraints.

(**Q3**) **Incorporation of different types of knowledge**. We studied the effectiveness of our unified framework in incorporating domain knowledge presented in heterogeneous forms. We considered knowledge in the form of CSI and MIS in 3 BN-based domains and 1 clinical domain. Table



(a) Varying γ_{size} (b) Varying γ_{noise} (c) Varying λ

Figure 4: **Ablation Study:** Mean test log-likelihood of a *RatSPN* trained on the Set-MNIST-Even dataset, across varying (a) size of domain sets and (b) degrees of noise. Shaded regions represent the SD across 3 trials.

3 presents the test log-likelihood scores of RatSPN learned from each of the 4 domains without knowledge (RatSPN), with a single CI encoded as CSIs (+CSI), and with the CSIs and a single MIS (+CSI+MIS). The models using both forms of knowledge perform better than models using CSIs.

(**Q4**) **Sensitivity and Robustness.** Our framework relies on two key hyperparameters: the size of the domain sets used for constraint evaluation (denoted by γ_{size}), and the penalty weight (λ) that balances constraint satisfaction with data-driven learning. In real-world applications, domain knowledge might be imperfect or noisy. We conducted ablation studies to assess the framework's sensitivity and robustness to these factors. γ_{size} is defined as the ratio of the domain set size for a constraint to the overall dataset size. To simulate noisy domain knowledge, we replaced a fraction of the domain set (γ_{noise}) with randomly sampled data points.

We investigated the impact of these hyperparameters on an EinsumNet model trained on the Set-MNIST-Even dataset with GC for 100 epochs. Varying γ_{size} (Figure 4a) showed that increasing the domain set size generally improves generalization performance, but this effect plateaus after a certain point (around $\gamma_{\text{size}} \approx 2$). Interestingly, varying γ_{noise} (Figure 4b) revealed that the model's performance remained relatively stable even with up to 40% noise in the constraints. This suggests the framework's robustness, as data compensates for some level of noise in the knowledge.

Finally, varying the penalty weight λ (Figure 4c) demonstrated that a very small λ leads to underutilization of the constraint, resulting in poor performance. Conversely, a very large λ overpowers the maximum likelihood training, disrupting the balance. The optimal λ appears to be around 1, striking a balance between data and constraints.

(**Q5**) **Performance on Real-World Data.** We evaluated the effectiveness of our framework on real-world data using the numom2b data sets. Tables 1 and 3 compare the test log-likelihood of *RatSPN* learned from the numom2b-a and numom2b-b data set with and without the knowledge in the form of MISs and CSIs. RatSPNs learned with knowledge achieved significantly higher test log-likelihood scores. This demonstrates the efficacy of our framework in exploiting domain knowledge to learn PCs from real-world datasets.

Discussion

We presented a general approach for incorporating domain knowledge as differentiable constraints into learning probabilistic models. It subsumes and extends several prior works on learning generative models (e.g., (Altendorf, Restificar, and Dietterich 2005; Papantonis and Belle 2021; Mathur, Gogate, and Natarajan 2023)) and (probabilistic) discriminative models (Kokel et al. 2020). Prior work by Papantonis and Belle (2021) focused specifically on knowledge encoded as equality constraints, which is a special case within our broader framework. We also generalize Mathur, Gogate, and Natarajan(2023)'s approach of using monotonicity constraints to learn cutset networks (CNs) both in the forms of knowledge that can be captured and the model class, as CNs can be represented as deterministic PCs.

Altendorf, Restificar, and Dietterich (2005) used *ceteris paribus*[1] monotonicity to learn the parameters of Bayesian Networks (BNs). Similar to CNs, BNs can be compiled into selective PCs (Peharz, Gens, and Domingos 2014), making our framework applicable here as well. The KiGB algorithm (Kokel et al. 2020) used soft monotonicity constraints for learning decision trees and decision trees can be converted into conditional SPNs (Shao et al. 2020). These methods can be seen as special cases of our framework.

Our framework is also versatile enough to incorporate more complex forms of knowledge such as (Odom et al. 2015; Xu et al. 2018), enhancing its applicability compared to previously discussed methods. Odom et al. (2015) used relational preference information to learn gradient-boosted relational trees. While our framework is limited to learning PCs with propositional logical constraints, it can be extended to use relational preference advice over variables. For example, the relational advice that "*Hypertensive disorders of pregnancy lead to pre-term birth*" could be compiled to the propositional *preference constraint* that subjects with the presence of any of chronic hypertension, gestational hypertension, and preeclampsia should have a higher conditional probability of pre-term birth. Xu et al. (2018) use knowledge in the form of propositional logical formulas to learn deep neural networks using a semantic loss that computes the log probability of satisfying the formulas. Since this probability can be computed tractably in a PC, the semantic loss can be used to learn PCs using our framework.

The simple yet effective objective function we propose in Eq. (5) has, in form, been successfully employed for tasks such as domain adaptation, domain generalization, and multiple class novelty detection (Sun and Saenko 2016; Shi et al. 2022; Perera and Patel 2019). We observed that augmenting the loss function with differentiable linear constraints is effective, as shown in our experiments, easy to be specified by domain experts, and allows the incorporation of a wide variety of constraints. While more complex objective functions are conceivable, our simple formulation encourages broader adoption. Integration of more complex forms of domain knowledge, handling structured data (relational data), as well as learning the structure of a PC in a knowledge-intensive manner are promising future directions.

[1]Other parents of the influenced variable X_i are kept constant

Acknowledgements

The authors gratefully acknowledge the generous support by the NIH grant R01HD101246, the AFOSR award FA9550-23-1-0239, the ARO award W911NF2010224, and the DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) award HR001122S0039.

References

Adel, T.; Balduzzi, D.; and Ghodsi, A. 2015. Learning the Structure of Sum-Product Networks via an SVD-based Algorithm. In *Conference on Uncertainty in Artificial Intelligence*.

Ahmed, K.; Teso, S.; Chang, K.-W.; Van den Broeck, G.; and Vergari, A. 2022. Semantic probabilistic layers for neuro-symbolic learning. *Advances in Neural Information Processing Systems*, 35: 29944–29959.

Altendorf, E.; Restificar, A. C.; and Dietterich, T. G. 2005. Learning from Sparse Data by Exploiting Monotonicity Constraints. In *UAI*, 18–26. AUAI Press.

Boutilier, C.; Friedman, N.; Goldszmidt, M.; and Koller, D. 1996. Context-Specific Independence in Bayesian Networks. In *UAI*.

Choi, Y.; Vergari, A.; and Van den Broeck, G. 2020. Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Models. *UCLA*.

Correia, A. H. C.; Gala, G.; Quaeghebeur, E.; de Campos, C.; and Peharz, R. 2023. Continuous Mixtures of Tractable Probabilistic Models. In *Proceedings of the AAAI Conference on Artificial Intelligence*.

Dang, M.; Vergari, A.; and Van den Broeck, G. 2020. Strudel: Learning Structured-Decomposable Probabilistic Circuits. In *Proceedings of the 10th International Conference on Probabilistic Graphical Models*, volume 138 of *Proceedings of Machine Learning Research*, 137–148. PMLR.

de Campos, C. P.; Tong, Y.; and Ji, Q. 2008. Constrained Maximum Likelihood Learning of Bayesian Networks for Facial Action Recognition. In *ECCV (3)*, volume 5304 of *Lecture Notes in Computer Science*, 168–181. Springer.

Deng, L. 2012. The MNIST database of handwritten digit images for machine learning research. *IEEE Signal Processing Magazine*.

Domingos, P. 1999. The role of Occam's razor in knowledge discovery. *Data mining and knowledge discovery*, 3: 409–425.

Galindez Olascoaga, L. I.; Meert, W.; Shah, N.; Van den Broeck, G.; and Verhelst, M. 2020. Discriminative bias for learning probabilistic sentential decision diagrams. In *Advances in Intelligent Data Analysis XVIII: 18th International Symposium on Intelligent Data Analysis, IDA 2020, Konstanz, Germany, April 27–29, 2020, Proceedings 18*, 184–196. Springer.

Gens, R.; and Domingos, P. 2013. Learning the structure of sum-product networks. In *International conference on machine learning*, 873–880. PMLR.

Ghandi, S.; Quost, B.; and de Campos, C. 2024. Probabilistic circuits with constraints via convex optimization. In *Joint European Conference on Machine Learning and Knowledge Discovery in Databases*, 161–177. Springer.

Haas, D. M.; Parker, C. B.; et al. 2015. A description of the methods of the Nulliparous Pregnancy Outcomes Study: monitoring mothers-to-be (nuMoM2b). *American journal of obstetrics and gynecology*.

Kim, J.; Yoo, J.; Lee, J.; and Hong, S. 2021. Setvae: Learning hierarchical composition for generative modeling of set-structured data. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 15059–15068.

Kokel, H.; Odom, P.; Yang, S.; and Natarajan, S. 2020. A Unified Framework for Knowledge Intensive Gradient Boosting: Leveraging Human Experts for Noisy Sparse Domains. In *AAAI*, 4460–4468. AAAI Press.

Korb, K. B.; and Nicholson, A. E. 2010. *Bayesian artificial intelligence*. CRC press.

Lauritzen, S. L. 1988. Local computations with probabilities on graphical structures and their application to expert systems (with discussion). *J. Roy. Statis., Soc., B*, 50(2): 251–263.

Li, J.; Xu, T.; Zhang, J.; Hertzmann, A.; and Yang, J. 2019. LayoutGAN: Generating Graphic Layouts with Wireframe Discriminator. In *International Conference on Learning Representations*.

Liu, A.; Zhang, H.; and den Broeck, G. V. 2023. Scaling Up Probabilistic Circuits by Latent Variable Distillation. In *The Eleventh International Conference on Learning Representations*.

Liu, X.; Liu, A.; Van den Broeck, G.; and Liang, Y. 2023. Understanding the distillation process from deep generative models to tractable probabilistic circuits. In *International Conference on Machine Learning*, 21825–21838. PMLR.

Luenberger, D. G.; and Ye, Y. 2016. *Linear and Nonlinear Programming*. Springer International Publishing.

Lüdtke, S.; Bartelt, C.; and Stuckenschmidt, H. 2022. Exchangeability-Aware Sum-Product Networks. In Raedt, L. D., ed., *Proceedings of the Thirty-First International Joint Conference on Artificial Intelligence, IJCAI-22*, 4864–4870. International Joint Conferences on Artificial Intelligence Organization.

Mathur, S.; Gogate, V.; and Natarajan, S. 2023. Knowledge Intensive Learning of Cutset Networks. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, 1380–1389. PMLR.

Mauro, N. D.; Vergari, A.; Basile, T. M. A.; and Esposito, F. 2017. Fast and Accurate Density Estimation with Extremely Randomized Cutset Networks. In *ECML/PKDD (1)*, volume 10534 of *Lecture Notes in Computer Science*, 203–219. Springer.

Odom, P.; Khot, T.; Porter, R.; and Natarajan, S. 2015. Knowledge-Based Probabilistic Logic Learning. In *AAAI*, 3564–3570. AAAI Press.

Papantonis, I.; and Belle, V. 2021. Closed-Form Results for Prior Constraints in Sum-Product Networks. *Frontiers Artif. Intell.*, 4: 644062.

Pasunuri, R.; Odom, P.; Khot, T.; Kersting, K.; and Natarajan, S. 2016. Learning with privileged information: Decision-trees and boosting. In *Proc. Int. Joint Conf. Artif. Intell. Workshop*, 1–7.

Peharz, R.; Geiger, B. C.; and Pernkopf, F. 2013. Greedy Part-Wise Learning of Sum-Product Networks. In Blockeel, H.; Kersting, K.; Nijssen, S.; and Železný, F., eds., *Machine Learning and Knowledge Discovery in Databases*, 612–627.

Peharz, R.; Gens, R.; and Domingos, P. 2014. Learning selective sum-product networks. In *LTPM workshop, ICML*, volume 32.

Peharz, R.; Lang, S.; Vergari, A.; Stelzner, K.; Molina, A.; Trapp, M.; den Broeck, G. V.; Kersting, K.; and Ghahramani, Z. 2020a. Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits. In *ICML*.

Peharz, R.; Vergari, A.; Stelzner, K.; Molina, A.; Shao, X.; Trapp, M.; Kersting, K.; and Ghahramani, Z. 2020b. Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning. In *UAI*.

Perera, P.; and Patel, V. M. 2019. Deep transfer learning for multiple class novelty detection. In *Proceedings of the ieee/cvf conference on computer vision and pattern recognition*, 11544–11552.

Plajner, M.; and Vomlel, J. 2020. Learning bipartite Bayesian networks under monotonicity restrictions. *Int. J. Gen. Syst.*, 49(1): 88–111.

Rooshenas, A.; and Lowd, D. 2014. Learning Sum-Product Networks with Direct and Indirect Variable Interactions. In *Proceedings of the 31st International Conference on Machine Learning*, volume 32 of *Proceedings of Machine Learning Research*, 710–718. Bejing, China: PMLR.

Rudin, C. 2019. Stop explaining black box machine learning models for high stakes decisions and use interpretable models instead. *Nat. Mach. Intell.*, 1(5): 206–215.

Rudin, C.; Zhong, C.; Semenova, L.; Seltzer, M.; Parr, R.; Liu, J.; Katta, S.; Donnelly, J.; Chen, H.; and Boner, Z. 2024. Amazing Things Come From Having Many Good Models. In *Proceedings of the International Conference on Machine Learning (ICML)*.

Sachs, K.; Perez, O.; Pe'er, D.; Lauffenburger, D. A.; and Nolan, G. P. 2005. Causal protein-signaling networks derived from multiparameter single-cell data. *Science*, 308(5721): 523–529.

Schwarz, G. 1978. Estimating the dimension of a model. *The annals of statistics*, 461–464.

Scutari, M.; and Denis, J.-B. 2021. *Bayesian networks: with examples in R*. Chapman and Hall/CRC.

Shao, X.; Molina, A.; Vergari, A.; Stelzner, K.; Peharz, R.; Liebig, T.; and Kersting, K. 2020. Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures. In *PGM*, volume 138 of *Proceedings of Machine Learning Research*, 401–412. PMLR.

Shi, Y.; Seely, J.; Torr, P.; N, S.; Hannun, A.; Usunier, N.; and Synnaeve, G. 2022. Gradient Matching for Domain Generalization. In *International Conference on Learning Representations*.

Sidheekh, S.; Dock, C. B.; Jain, T.; Balan, R.; and Singh, M. K. 2022. VQ-Flows: Vector quantized local normalizing flows. In Cussens, J.; and Zhang, K., eds., *Proceedings of the Thirty-Eighth Conference on Uncertainty in Artificial Intelligence*, volume 180 of *Proceedings of Machine Learning Research*, 1835–1845. PMLR.

Sidheekh, S.; Kersting, K.; and Natarajan, S. 2023. Probabilistic Flow Circuits: Towards Unified Deep Models for Tractable Probabilistic Inference. In *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216 of *Proceedings of Machine Learning Research*, 1964–1973. PMLR.

Sidheekh, S.; and Natarajan, S. 2024. Building Expressive and Tractable Probabilistic Generative Models: A Review. In *IJCAI*, 8234–8243. Survey Track.

Sun, B.; and Saenko, K. 2016. Deep coral: Correlation alignment for deep domain adaptation. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part III 14*, 443–450. Springer.

Towell, G. G.; and Shavlik, J. W. 1994. Knowledge-Based Artificial Neural Networks. *Artif. Intell.*, 70(1–2): 119–165.

van der Gaag, L. C.; Bodlaender, H. L.; and Feelders, A. J. 2004. Monotonicity in Bayesian Networks. In *UAI*, 569–576. AUAI Press.

Ventola, F.; Braun, S.; Zhongjie, Y.; Mundt, M.; and Kersting, K. 2023. Probabilistic circuits that know what they don't know. In Evans, R. J.; and Shpitser, I., eds., *Proceedings of the Thirty-Ninth Conference on Uncertainty in Artificial Intelligence*, volume 216, 2157–2167. PMLR.

Xu, J.; Zhang, Z.; Friedman, T.; Liang, Y.; and den Broeck, G. V. 2018. A Semantic Loss Function for Deep Learning with Symbolic Knowledge. In *ICML*, volume 80 of *Proceedings of Machine Learning Research*, 5498–5507. PMLR.

Yang, S.; Khot, T.; Kersting, K.; Kunapuli, G.; Hauser, K.; and Natarajan, S. 2014. Learning from imbalanced data in relational domains: A soft margin approach. In *ICDM*.

Yang, S.; and Natarajan, S. 2013. Knowledge Intensive Learning: Combining Qualitative Constraints with Causal Independence for Parameter Learning in Probabilistic Models. In *ECML/PKDD (2)*, volume 8189 of *Lecture Notes in Computer Science*, 580–595. Springer.

Zaheer, M.; Kottur, S.; Ravanbakhsh, S.; Poczos, B.; Salakhutdinov, R. R.; and Smola, A. J. 2017. Deep Sets. In Guyon, I.; Luxburg, U. V.; Bengio, S.; Wallach, H.; Fergus, R.; Vishwanathan, S.; and Garnett, R., eds., *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Zhang, Y.; Hare, J.; and Prugel-Bennett, A. 2019. Deep set prediction networks. *Advances in Neural Information Processing Systems*, 32.