Building Expressive and Tractable Probabilistic Generative Models: A Review

Sahil Sidheekh, **Sriraam Natarajan**

The University of Texas at Dallas

{sahil.sidheekh, sriraam.natarajan}@utdallas.edu,

Abstract

We present a comprehensive survey of the advancements and techniques in the field of tractable probabilistic generative modeling, primarily focusing on Probabilistic Circuits (PCs). We provide a unified perspective on the inherent trade-offs between expressivity and tractability, highlighting the design principles and algorithmic extensions that have enabled building expressive and efficient PCs, and provide a taxonomy of the field. We also discuss recent efforts to build deep and hybrid PCs by fusing notions from deep neural models, and outline the challenges and open questions that can guide future research in this evolving field.

1 Introduction

Generative modeling plays an important role in the fields of machine learning and artificial intelligence, as it offers a powerful toolkit for understanding, interpreting as well as recreating complex patterns present in our data-rich world. By employing probability theory as a principled way to capture the inherent uncertainty in a given dataset, these models aim to approximate the underlying distribution or random process responsible for generating the data. Consequently, probabilistic generative models possess the potential to solve a diverse range of problems, including generating new data samples, performing inference given observations, estimating likelihood of events, and reasoning about uncertain information.

However, learning the distribution back from data is a challenging problem that often necessitates trade-offs between modeling flexibility and the tractability of probabilistic inference. Early generative models prioritized enabling tractable inference, often by imposing probabilistic structures over random variables, in the form of graphical models [Koller and Friedman, 2009]. However, as a result, they lacked the flexibility to model complex distributions. The field of *Tractable Probabilistic Models* (TPMs) have since evolved, with expressive parameterizations and learning paradigms proposed, resulting in a broad and popular class of models under the unified notion of *probabilistic circuits*. Designed from the perspective of tractability, these models enable efficient inference and exact probabilistic reasoning, making them suitable for tasks demanding fast and exact computations. However,

they still struggle to capture dependencies when the data complexity and dimensionality increase.

In contrast, advancements in deep learning have given rise to expressive Deep Generative Models (DGMs) that exploit the power of neural networks to learn flexible representations of complex data distributions. Notable examples include generative adversarial networks, variational autoencoders, normalizing flows and diffusion models. Prioritizing expressiveness, these models have demonstrated impressive proficiency in capturing complex dependencies and generating high fidelity samples. However, unlike TPMs, they often lack the ability to reason explicitly about the learned distribution.

Bridging the gap between TPMs and DGMs is thus a fascinating area of research, aiming to combine their strengths and create hybrid models that are expressive as well as tractable. This survey aims to comprehensively explore techniques and recent advancements in this direction. While previous surveys have extensively studied DGMs [Bond-Taylor *et al.*, 2022] and TPMs [Sánchez-Cauce *et al.*, 2021] independently, analyzing their design principles and associated challenges, a unified and cohesive view is still lacking. Through this work, we thus aim to fill this gap and provide researchers with a holistic understanding of the field. We hope to highlight the benefits and challenges of this synergistic combination to motivate enhanced research in this direction.

We will begin by discussing the building blocks, properties, learning methodologies as well as challenges when building tractable generative models, focusing on probabilistic circuits, and provide a broad taxonomy of the field. We will then discuss hybrid techniques that merge TPMs with DGMs to achieve the best of both worlds. Finally, we will identify challenges, open problems, and potential directions that can lay the foundations for future research in this field. We aim to serve the dual purpose of providing beginners interested in the field with a broad understanding of how to build expressive and tractable generative models, while empowering experienced researchers to push the boundaries by understanding the intricacies and complexities of the domain.

Notation. We use $\mathbf{X} = (X_1, X_2, \ldots, X_d)$ to denote a set of $d-$ random variables and $\mathbf{x} = (x_1, x_2, \ldots, x_d)$ denote an assignment of values to $\mathbf{X}$. We use $P_{\mathbf{X}}$ to denote the true probability distribution over $\mathbf{X}$ and P_θ to denote the distribution captured by a generative model parameterized by θ.

2 Tractable Probabilistic Modeling

Given a dataset $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^M$ of M instances, each comprising d features, the generative modeling problem can be formalized as learning a model θ that best explains $\mathcal{D}$, i.e. finding an optimal θ^* such that $P_{\theta^*} \approx P_{\mathbf{X}}$. The optimization problem can be formulated using maximum likelihood as: $\theta^* = \arg\max_{\theta} P_{\theta}(\mathcal{D}) = \arg\max_{\theta} \prod_{i=1}^M P_{\theta}(\mathbf{x}^{(i)})$.

A probabilistic generative model is said to be *tractable* for an inference task, if the answer to the corresponding probabilistic query can be computed *exactly* in time that is polynomial with respect to the size of the model and the input. Thus, tractability is a spectrum that depends not only on the characteristics of the generative model, but also on the type of probabilistic query, i.e. the inference task. As we will see, some inference tasks are inherently harder than others and a model tractable for one query might not be for another.

2.1 Inference Queries

We begin by outlining some of the prevalent inference tasks that we might be interested in performing, in order to make decisions using a probabilistic generative model effectively. The most frequent and basic inference scenario typically involves calculating the probability associated with a specific assignment of values to all the random variables. This is commonly referred to as **evidential inference** and involves computing $P_{\theta}(\mathbf{X} = \mathbf{x})$ for a given assignment $\mathbf{x}$ exactly, without resorting to any approximations. In many real world problems, we are interested in reasoning about only a subset of the modeled variables due to the non-homogenous nature of the data or missing features. Or sometimes, in the presence of sensitive features, enforcing fairness in decision-making might require us to marginalize out the effect of certain variables. More formally, given subsets $\mathbf{X}_1, \mathbf{X}_2$ such that $\mathbf{X}_1 \cup \mathbf{X}_2 = \mathbf{X}$ and $\mathbf{X}_1 \cap \mathbf{X}_2 = \emptyset$, evaluating the likelihood of the subset of variables of interest (say $\mathbf{X}_1$) by marginalizing out rest is referred to as **marginal inference**, i.e. it involves computing $P_{\theta}(\mathbf{X}_1) = \int_{\mathbf{X}_2} P_{\theta}(\mathbf{X}_1, \mathbf{X}_2) d\mathbf{X}_2$. Marginal inference involves performing evidential inference as a subroutine a large (possibly infinite) no. of times, and is thus a harder task than the latter. The third type of inference scenario involves computing the probability of an event A, given that another event B has happened. This is referred to as **conditional inference**, and it involves computing $P_{\theta}(\mathbf{X}_1|\mathbf{X}_2) = \frac{P_{\theta}(\mathbf{X}_1, \mathbf{X}_2)}{P_{\theta}(\mathbf{X}_2)} = \frac{P_{\theta}(\mathbf{X}_1, \mathbf{X}_2)}{\int_{\mathbf{X}_2} P_{\theta}(\mathbf{X}_1, \mathbf{X}_2) d\mathbf{X}_2}$.

Maximum-a-Posteriori (MAP) inference seeks the most probable variable assignments based on evidence, essentially maximizing the posterior distribution over the variables of interest. This task is crucial for parameter estimation and predicting the most likely outcomes. More formally, it invoves computing $\arg\max_{\mathbf{X}_1} P_{\theta}(\mathbf{X}_1|\mathbf{X}_2) = \arg\max_{\mathbf{X}_1} P_{\theta}(\mathbf{X}_1, \mathbf{X}_2)$.

2.2 Enforcing Tractability by Imposing Structure

A prevalent strategy to attain tractability in probabilistic inference for generative models is the imposition of structural constraints within them. For instance, considering all random variables as independent leads to a highly structured model: $P_{\theta}(\mathbf{X}) = \prod_{i=1}^d P_{\theta}(X_i)$. This simplifies the computation of marginals, conditionals, and MAP, reducing them to operations over univariate distributions. Incorporating such factorizations of the joint distribution is thus key for achieving tractability. However, this often limits the model's expressiveness, as more restrictive assumptions reduce the variety of representable probability distributions. To balance flexibility and simplicity, *mixture models*, which are convex combinations of simpler distributions, are typically employed. A notable example is the Gaussian Mixture Model, which is theoretically capable of approximating any continuous distribution in the limit, given adequate components [Lindsay, 1995]. Thus, designing algorithms that enforce structure via factorizations and flexibility via mixtures is an effective approach to build tractable models without sacrificing expressivity.

3 Probabilistic Circuits

Building on the above principle of factorizations and mixtures, several classes of tractable probabilistic models have emerged, such as Arithmetic Circuits [Darwiche, 2003], Probabilistic Sentential Decision Diagrams [Kisa *et al.*, 2014], AND-OR search spaces [Marinescu and Dechter, 2005], Sum Product Networks [Poon and Domingos, 2011], Cutset Networks [Rahman *et al.*, 2014], etc. Recently, [Choi *et al.*, 2020] presented a unified view of such models, generalizing them under the umbrella notion of *Probabilistic Circuits* (PCs). We emphasize on PCs as a tractable representation for learning data distributions **but go beyond their work by providing a unified view of the different models, their learning methodologies, and presenting the latest work in employing ideas from deep learning**.

Definition 1. *A Probabilistic Circuit $\mathcal{C}$ is a computational graph that compactly encodes a probability distribution via factorizations and mixtures. It consists of three types of nodes - Sums, Products and Leaf Distributions. Each node in the graph computes a non-negative function, which can be interpreted as an un-normalized probability measure over a subset of random variables, which is referred to as the scope of the node. The computational graph is evaluated bottom up and is recursively defined as follows:*

- *For a sum node $(+)$ with scope $\psi_+ \subseteq \mathbf{X}$, the output is defined as a convex combination of the outputs of its children. i.e $+(\mathbf{x}_{\psi_+}) = \sum_{N_i \in child(+)} w_i N_i(\mathbf{x})$, where $w_i \geq 0$ and $\sum_i w_i = 1$.*

- *For a product node $(\times)$ with scope $\psi_\times \subseteq \mathbf{X}$, the output is defined as a product of the outputs of its children. i.e $\times(\mathbf{x}_{\psi_\times}) = \prod_{N_i \in child(\times)} N_i(\mathbf{x}_{\psi_{N_i}})$.*

- *A leaf node L represents a simple tractable univariate distribution over its scope $\psi_L \subseteq \mathbf{X}$, such as a Gaussian, and its output is defined as the probability density (or mass for discrete variables) under the corresponding distribution. $L(\mathbf{x}_{\psi_L}) = P_L(\mathbf{x}_{\psi_L})$*

The output of the root node constitutes the modeled density.

A probabilistic circuit $\mathcal{C} = \{\mathcal{G}, \theta\}$ thus has both a structure encoded as a computational graph $\mathcal{G}$ and parameters θ which
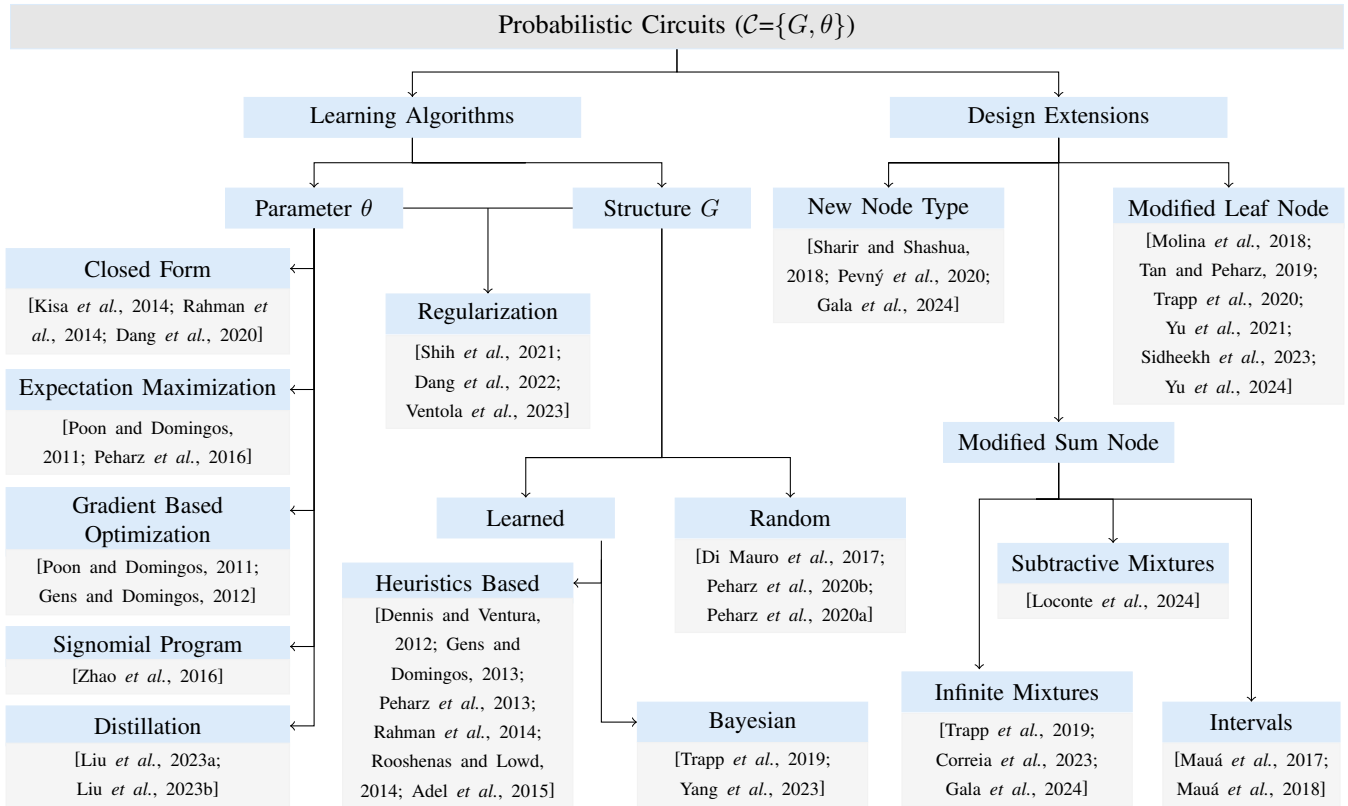
Figure 1: A broad taxonomy of the literature on improving probabilistic circuits via better learning algorithms and design extensions.

corresponds to the weights associated with the sum nodes and the parameters of the leaf distribution. It can be viewed as a hierarchical mixture model that compactly encodes an exponential no. of factorized mixture components.

3.1 Structural Properties for Probabilistic Circuits

In order to ensure that $\mathcal{C}$ models a valid distribution and supports tractability, we need to define further structural properties over them, which we elaborate below.

Definition 2 (Smoothness). *A probabilistic circuit is said to be **smooth**, if all of it's sum nodes are defined over children having the same scope.*

Definition 3 (Decomposability). *A probabilistic circuit is said to be **decomposable**, if all of it's product nodes are defined over children having disjoint scope.*

Definition 4 (Determinism). *A probabilistic circuit is said to be **deterministic**, if, for all of its sum nodes, the output of at most one of its children is non-zero for any given input.*

Intuitively, smoothness implies that the sum nodes represent valid mixture distributions. This, in turn implies the tractability of evidential inference, as the data density can be computed by evaluating the circuit bottom up, which can be done in time linear in the size of the circuit. Further, circuits that are smooth and decomposable additionally support tractable computation of marginal and conditional queries. This is because the integrals involved in these queries decomposes over the sum and product nodes to their children.

Thus for a smooth and decomposable PC, we can push down the integrals recursively until we reach the leaves. Since the leaf distributions are simple and tractable, the integrals can be computed analytically. Evaluating the marginal or conditional query then reduces to performing a bottom-up pass on the circuit, with the values of the leaf nodes set to their corresponding integrals, and is thus also linear in the size of the circuit. However, smoothness and decomposability are not sufficient for tractably computing MAP queries. This is because the maximizer over a convex combination of distributions is not necessarily the convex combination of the maximizers of the individual distributions. However, we can perform MAP inference tractably over smooth, decomposable, and *deterministic* PCs. Note that deterministic PCs are also sometimes called *selective* as the sum nodes can be viewed as selecting one of its children [Peharz *et al.*, 2014].

3.2 Parameter Learning

As PCs support density evaluation, we can employ maximum likelihood to learn their parameters. In the presence of determinism, the output of the root node reduces to a weighted product of simple factorized distributions, and the parameters can be estimated in closed form [Rahman *et al.*, 2014]. However, in the more general case of smooth and decomposable circuits, a closed form solution is not available, and we will have to resort to iterative optimization schemes to maximize the data likelihood.

Gradient Based Optimization. Since PCs are defined as

computational graphs, they are differentiable, and the partial derivative of the output of the root node (which represents the data density) w.r.t to each node in the circuit can be easily computed. Thus, initializing the model parameters randomly, these gradients can then be used to iteratively update the parameters to maximize the likelihood and learn the data distribution. In practice, one could implement these computational graphs using packages that support automatic differentiation, and backpropagate the gradients efficiently. While one can employ full batch gradient ascent to optimize this objective, its stochastic version that samples mini batches is more compute efficient and fast, and is popularly used for learning PCs in the generative and well as discriminative settings [Poon and Domingos, 2011; Gens and Domingos, 2012]. More complex gradient-based optimizers such as Adam that incorporates momentum have also been used to learn PCs [Peharz *et al.*, 2020b; Peharz *et al.*, 2020a; Sidheekh *et al.*, 2023].

Expectation Maximization. An alternative optimization scheme proposed for PCs is expectation-maximization (EM), which is popularly used for maximum likelihood learning in the presence of missing data [Dempster *et al.*, 1977]. The crux of EM involves iterating between the following two steps until convergence, after initializing from a random configuration - (1) The E-Step, which involves computing the expected value of the missing variables given the observed variables and (2) the M-step, which maximizes the likelihood given the complete expected assignment. As PCs are essentially hierarchical mixture models, their sum nodes can be viewed as marginalizing out an unobserved discrete latent variable. [Peharz *et al.*, 2016] formalized this latent variable interpretation for PCs by explicitly introducing the unobserved latent variables in the computational graph to create an *augmented* circuit. As we do not know the assignments for these variables for the data points in hand, it becomes a learning with missing data problem and can be solved using expectation maximization. Compared to gradient ascent, EM has been observed to result in a larger boost in the data likelihood [Peharz *et al.*, 2016; Peharz *et al.*, 2020b; Peharz *et al.*, 2020a] especially in the early phase of learning.

[Zhao *et al.*, 2016] presented a unified view of parameter learning in smooth and decomposable PCs. They observed that the MLE optimization problem can be formulated as a signomial program and proposed two algorithms for parameter learning - (1) sequential monomial approximations (SMA) that generalized gradient descent and (2) the concave-convex procedure (CCP) that generalized expectation maximization. All of the approaches discussed above focus on data-driven parameter learning, which can be challenging when the data is *noisy* and *sparse*. Knowledge-intensive learning of PCs [Mathur *et al.*, 2023; Karanam *et al.*, 2024] has been proposed as a more robust framework under such settings.

3.3 Structure Learning

The computational graph structure $\mathcal{G}$ of a PC encodes the factorizations of the joint distribution, and hence can impact its expressivity. However, the optimal structure can vary for each data distribution and is seldom known apriori.

Heuristics Based. Learning the graph structure from data

was first explored by [Dennis and Ventura, 2012]. They observed that the latent variable associated with a sum node should help explain interactions between variables in its scope. Hence they proposed to strategically locate sum nodes over groups of variables that have pronounced interdependencies via clustering. Subsequently, [Gens and Domingos, 2013] showed that the above apporach does not make use of context specific independencies and is prone to splitting highly dependent variables into different clusters, thus causing a large loss of likelihood. Instead they proposed a greedy iterative algorithm, known as LearnSPN that could be used to define the scopes for both the sum nodes as well as product nodes. Specifically, at a product node, they employed statistical independence testing to identify mutually independent subsets of variables. And at a sum node, they employed EM based clustering to group similar instances as the support for each of its child. The weight for a sum node edge could now be defined as the fraction of data points belonging to its corresponding cluster. Recursively employed, this algorithm resulted in the extraction of sets of columns from the data matrix at a product node and sets of rows at a sum node until a univariate leaf was reached. Building further, [Rooshenas and Lowd, 2014] proposed to merge the indirect interactions modeled by the latent variables clustered at higher levels, with direct interactions of observed variables by employing tractable Markov networks at the lower levels. Similarly, [Adel *et al.*, 2015] proposed an SVD based structure learning algorithm that merged the row-wise and column-wise splitting of the data matrix employed by LearnSPN into a single operation of extracting rank-one submatrices. Other approaches have explored incorporating information bottleneck [Peharz *et al.*, 2013] as well as cutset-conditioning that mimicks decision tree learning [Rahman *et al.*, 2014]. However all of the above methods are based on heuristics and lacks solid theoretical grounding.

Bayesian Approaches. Perhaps the most principled and elegant approach to structure learning for PCs is to adopt a Bayesian view. By viewing the parameterized density as a function of both θ and $\mathcal{G}$, i.e. $P_{\theta,\mathcal{G}}(\mathbf{x}) = P(\mathbf{x}|\theta,\mathcal{G})$, we can define *Bayesian Structure Score* ($\mathcal{B}$) as the contribution of $\mathcal{G}$ to the overall likelihood, i.e.: $\mathcal{B}(\mathcal{G}) = P(\mathcal{D}|\mathcal{G}) = \int_\theta P(\mathcal{D}|\theta,\mathcal{G})P(\theta|\mathcal{G})d\theta = \int_\theta P(\theta|\mathcal{G})\prod_{\mathbf{x}\in\mathcal{D}} P(\mathbf{x}|\theta,\mathcal{G})d\theta$
Intuitively, this reduces to assuming a prior distribution $P(\theta|\mathcal{G})$ over the parameters and computing the contribution of the structure $\mathcal{G}$ to the likelihood by marginalizing out the parameters. Such a score can then be used to optimize over structures in a Bayesian way [Friedman and Koller, 2003] by employing search algorithms [Russell, 2010] or even structural expectation maximization [Friedman, 1998]. However, computing the above score is non-trivial as it involves integrating over parameters. Recently, [Yang *et al.*, 2023] showed that this score can be computed tractably and exactly for deterministic PCs. They employed the structure score together with the greedy cutset learning algorithm [Rahman *et al.*, 2014] as well as structural EM to learn state of the art PCs. They also showed that when the data is discrete, and the prior over the sum node parameters is assumed to be a Dirichlet distribution, the Bayesian Structure Score reduces to the well known Bayes-Dirichlet (BD) score [Heckerman *et al.*, 1995].

3.4 Deep PCs via Random Structures

Tangential to the above discussed direction of enhancing probabilistic circuits by learning their structure from data, there exists a diverse range of approaches that adopt the perspective that **structure may not be highly relevant when you have the ability to over-parameterize**. [Peharz *et al.*, 2020b] showed that by utilizing a sufficiently large ensemble of random structures, comparable performance can be achieved as with a learned structure. To create valid random structures, they extended the notion of Region Graphs introduced in [Dennis and Ventura, 2012; Peharz *et al.*, 2013] to Random Region Graphs. Intuitively, a random region graph over a set of variables can be viewed as a rooted directed acyclic graph that recursively and randomly partitions the variables associated with each node. The region graph can be converted into a valid tensorized probabilistic circuit, called RAT-SPN [Peharz *et al.*, 2020b] by populating them with arrays of sum nodes, product nodes and leaf distributions. Unlike classical parameterizations, the computational graph for the above parameterization has reduced sparsity, and is well amenable to GPU level-parallelization, making it highly scalable. Along similar lines, efficiently utilizing randomized structures for the specific case of deterministic probabilistic circuits was also explored by [Di Mauro *et al.*, 2017; Mauro *et al.*, 2021]. Notably, [Peharz *et al.*, 2020a] extended the RAT-SPN framework by introducing a novel implementation design. They combined the vectorized sum and product operations into a single monolithic *einsum* operation. This enabled designing PCs by stacking einsum layers similar to a deep neural network, allowing for even more parallelized computation, resulting in up to two orders of magnitude improvements in training times. Extending further, [Mari *et al.*, 2023] presented a unified framework that encompasses multiple such tensorized parameterizations for PCs, and showed that low-rank decompositions can be employed to improve their computational efficiency, while retaining expressivity.

Probabilistic Circuits vs Neural Networks. As PCs are computational graphs, they are similar in essence to neural networks. The random parameterizations discussed above have enabled building deep PCs, bringing them closer to deep neural models. However, it is important to note that they still differ from neural networks on multiple aspects. Most importantly, the computational graph of a PC transforms probability densities associated with the data, unlike neural networks (and DGMs) which transforms the data itself. Further, the parameters associated with a PC have probabilistic semantics and as a result more structure in contrast to unconstrained parameters typical in the context of neural networks.

3.5 Extensions and Modifications

Several attempts have been made to extend the definition of PCs for improving their expressivity and robustness. [Sharir and Shashua, 2018] proposed the introduction of quotient nodes representing conditional distributions within PCs. They showed that the resulting class of models are more expressive, while still capable of tractable inference. [Trapp *et al.*, 2019] generalized sum nodes in a PC to mixtures with an infinite number of components. [Mauá *et al.*, 2017; Mauá *et al.*, 2018] proposed replacing the scalar sum weights

in a PC with intervals, resulting in robust circuits that represent a credal set of distributions. [Loconte *et al.*, 2024] relaxed the non-negativity assumption on the sum node weights in a PC to build deep *subtractive mixture models*. They showed that by squaring such models, we can construct valid PCs that are exponentially more expressive than their additive counterparts with convex sum node weights.

Other approaches have tried to improve the expressivity by introducing more flexible leaf distributions. [Molina *et al.*, 2018] proposed the use of piecewise polynomial leaf distributions. Similarly, [Trapp *et al.*, 2020] proposed the integration of gaussian processes at the leaves, however its added expressivity came at the cost of tractability. [Yu *et al.*, 2024] proposed the use of univariate characteristic functions in the spectral domain as leaves in a PC to construct *characteristic circuits*. They showed that the resulting model could better capture heterogenous data distributions that do not have closed form probability density functions. Figure 1 provides a systematic categorization of the various methodologies discussed above for improving PCs via better learning algorithms and design extensions.

4 Building Bridges between DGMs and PCs

With efficient and scalable deep parameterizations available for learning PCs, as outlined above, it is natural to assume that we can improve their expressive power by building larger overparameterized models. However, [Liu *et al.*, 2023a] demonstrated that scaling the parameters of a PC does not result in a corresponding performance improvement. PCs are still **far from achieving the expressivity of DGMs**. Thus, there is a growing interest in fusing concepts and inductive biases from deep generative models within PCs, to build hybrid models that can balance the expressive power of DGMs with the computational tractability, robustness, stability, and interpretability of PCs, which we outline next.

4.1 Probabilistic Circuits with Neural Networks

One of the prevalent reasons that contributed to the popularity of deep learning was its ability to incorporate inductive biases such as translation invariance for image data effectively by employing convolutions. [Butz *et al.*, 2019] demonstrated that the sum nodes in a PC are similar in essence to convolutions, while product nodes resemble pooling operations employed by deep neural models. They formalized the properties that such neural operations need to satisfy in order to result in a valid PC, thus building a class of hybrid and deep yet tractable convolutional PCs. Along similar lines, [Yu *et al.*, 2022a] proposed the integration of the *self attention mechanism*, that has made transformer based models popular, within PCs, while [Ventola *et al.*, 2020] proposed the use of residual links, thus developing a probabilistic analog to ResNets [He *et al.*, 2016]. [Shih *et al.*, 2021] proposed the use of neural networks for making PCs robust to overfitting. They partitioned the sum node weights of a PC into multiple *sectors*, learned a lower dimensional embedding for each sector, and used small neural networks to map the embeddings to the parameters of the PC. This can be viewed as a soft weight-sharing mechanism where multiple parameters are generated
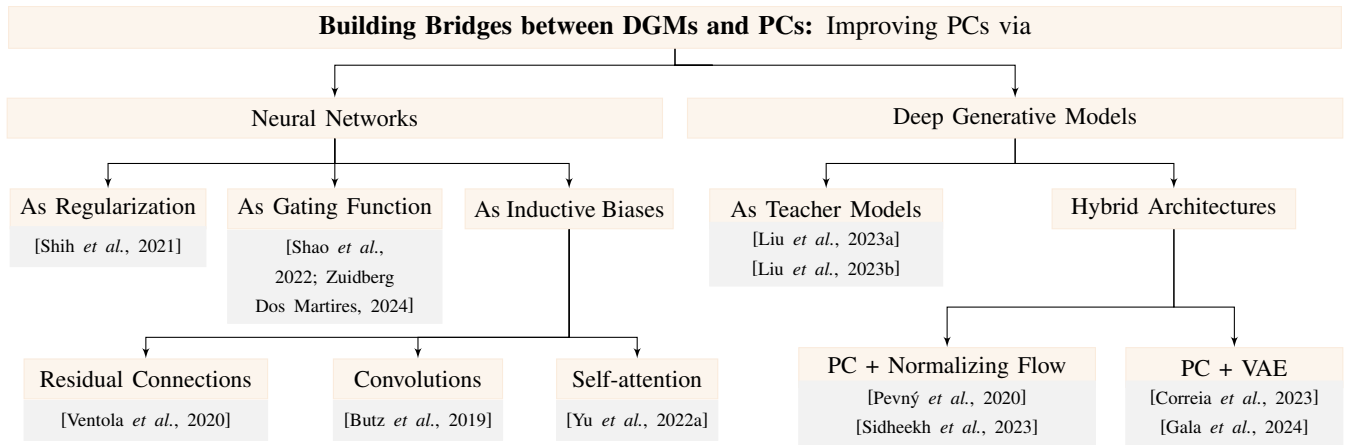
Figure 2: A taxonomy of the literature on building hybrid models to bridge the gap between deep generative models and probabilistic circuits

by a single neural network. The resulting PC thus has reduced degrees of freedom and better generalization. [Shao *et al.*, 2022] considered conditional PCs for structured output prediction tasks, which can be viewed as modeling the conditional distribution $P_\theta(\mathbf{Y}|\mathbf{X})$ over a set of targets $\mathbf{Y}$ and features $\mathbf{X}$, and proposed integrating neural networks as gating functions. More specifically, they computed the mixture weights of the PC as a function of the input features $\mathbf{X}$ via a neural network. They demonstrated that the resulting model not only had added expressivity while retaining tractable inference capabilities over the target variables $\mathbf{Y}$, but could also be used to effectively impose structure over DGMs. More recently, [Zuidberg Dos Martires, 2024] extended conditional PCs capturing factorizations of the joint distribution as given by a partial order ψ defined over the set of random variables $\mathbf{X}$ by generalizing the scalar weights associated with a sum node, say defined over $X \in \mathbf{X}$, to be a neural network transformation of the ancestors of X in ψ. The resulting model, called *probabilistic neural circuits*, can be viewed as deep, hierarchical mixtures of Bayesian Networks.

4.2 Probabilistic Circuits with VAEs

[Liu *et al.*, 2023a] attributed the failure of PCs in the overparameterized regime to the increase in latent information (associated with sum nodes) as PCs are scaled, which in turn made the marginal likelihood over observed variables more complex and hence maximum likelihood training more challenging. To address this issue, they proposed to provide extra supervision to PC learning by explicitly materializing the latent variables using a less tractable but more expressive deep generative model. As VAEs are effective models for learning latent representations, [Liu *et al.*, 2023a] utilized Masked Autoencoders [He *et al.*, 2022] to learn feature representations for the sum nodes and employed K-means clustering in this feature space to obtain the assignments for the discrete latent variables associated with sum nodes. The resulting framework, which they called Latent Variable Distillation (LVD) was able to achieve competitive performance against widely used DGMs. Building further, [Liu *et al.*, 2023b] studied the theoretical properties as well as design principles of

the DGMs to be used as teacher models for LVD. They observed that when performing LVD, the performance of the student PC can exceed that of the teacher DGM. They also highlighted that the disparity between the continuous latent representations learned by DGMs and the discrete latent variable assignments demanded by a PC can lead to information loss. They proposed to overcome this issue by employing a progressive growing algorithm that leveraged feedback from the PC to perform dynamic clustering.

VAEs can be interpreted as mixture models with an infinite number of components, where the components depend continuously on the latent code, and are thus intractable. PCs on the other hand are discrete hierarchical mixture models. [Correia *et al.*, 2023] observed that even a huge discrete mixture model such as an overparameterized PC is unable to outperform a relatively moderately sized uncountable mixture model such as a small VAE, suggesting that continuous mixtures generalize better or are easier to learn as opposed to PCs. Thus, they proposed to merge VAEs with PCs by taking continuous mixtures of tractable PCs. Their approach can be intuitively understood as replacing the decoder of a VAE with a PC. Though the resulting formulation is intractable in practice, the authors demonstrated that it can be approximated arbitrarily well using numerical integration techniques when the considered latent space has a low dimension. [Gala *et al.*, 2024] further generalized this approach by allowing integral units representing continuous latent variables to be defined not only at the root but also as internal nodes in a PC.

In the reverse direction, improving VAEs by using PCs was explored by [Tan and Peharz, 2019], who built hierarchical mixtures of VAEs by employing them at the leaf nodes of a PC. Though the resulting model lacked tractability, the authors showed that such a hybrid model resulted in improved stability and better learning of VAEs.

4.3 Probabilistic Circuits with Normalizing Flows

Normalizing flows constitute one of the most structured class of DGMs that utilize diffeomorphic neural transformations to map a simple base distribution into a more complex one. Their diffeomorphic structure enables computing the proba-

bility density exactly using the change of variables formula, and are thus tractable models for evidential inference. Naturally, extending PCs by employing the change of variables principle of flows has been explored in the literature. To achieve this, [Pevný *et al.*, 2020] proposed the addition of a new node type - called transform nodes, arbitrarily over existing nodes in a PC. Each transform node ($\mathcal{T}$) was associated with an invertible affine transformation and defined over a single child node. The output of $\mathcal{T}$ was defined recursively as the transformation of the distribution modeled by its child.

Building further, [Sidheekh *et al.*, 2023] showed that the above construction for integrating normalizing flows with PCs can violate the decomposability of PCs, hence making complex inference queries such as marginals and conditionals intractable. They formalized the necessary conditions that transform nodes need to satisfy in order to retain tractability, which they called $\tau-$decomposability. Intuitively, $\tau-$decomposability demanded that when $\mathcal{T}$ is defined over a product node $\mathcal{P}$, it should transform the scopes of the children of $\mathcal{P}$ independently. They also showed that integrating $\tau-$decomposable transform nodes arbitrarily in a circuit is equivalent to defining normalizing flows over the leaf distributions of a PC. They demonstrated that utilizing invertible linear rational spline transformations at the leaves, we can build expressive yet tractable *probabilistic flow circuits*.

4.4 Implications of Expressive & Tractable Models

From Approximations to Exactness. The tractability offered by PCs can be utilized to solve classical problems solved via approximations *exactly*. [Shih and Ermon, 2020] studied PCs in the context of discrete graphical models, and showed that they can be employed as expressive variational families that support exact ELBO computation as well as stable gradients. [Khosravi *et al.*, 2019] showed that PCs can be used to compute expectations as well as higher order moments of the predictions of discriminative models.

To More Complex Tasks. [Choi *et al.*, 2022] showed that the complex inference routine of Marginal MAP can be solved exactly by leveraging PC transformations. [Ventola *et al.*, 2023] showed that Monte Carlo Dropout, when introduced within the context of PCs and be used for uncertainty quantification exactly and efficiently, thus making PCs robust to out of distribution data. [Selvam *et al.*, 2023] utilized the tractability of PCs to reason with partial data to search for discrimination patterns and certify the model's *fairness*.

[Vergari *et al.*, 2021] compiled a comprehensive catalogue of operations invloving probability distributions that can be tractably computed using PCs, outlining the structural properties that needs to be satisfied within the context of each operation. Their work generalized the common inference queries we have seen so far to also include the computation of sums, products, quotients, powers, logarithms and exponentials of probability distributions encoded as PCs. As a result, complex information theoretic quantities such as divergence measures, which typically demand approximations, could be represented via tractable and modular operations over circuits.

Broader Applications Across Fields. PCs, as expressive and tractable models, have found applications across various domains spanning causality, neuro-symbolic learning, and controlled generation. Notably, [Wang and Kwiatkowska, 2023] showed that compositions of probabilistic operators defined via PCs can be used to answer causal inference queries such as the *backdoor adjustment* in polynomial time. [Liu *et al.*, 2022] showed that PCs can be adapted to perform lossless compression effectively. [Ahmed *et al.*, 2022] showed that PCs can be used to restrict the support of deep classifier outputs, serving as an effective interface to integrate symbolic knowledge with deep neural models. Further, PCs have also been employed to effectively control the generative process in DGMs such as autoregressive language models [Zhang *et al.*, 2023] and diffusion models [Liu *et al.*, 2024].

5 Open Problems and Promising Directions

We have so far covered the various algorithms and design principles that has enabled building expressive and tractable generative models. However, the field has several open problems, which makes it a fertile area for further research and significant advancements in several directions.

Theory of Optimizing Overparameterized PCs. Most works that attempt to overcome the performance plateau of overparameterized PCs are based on heuristics. A theoretically grounded understanding of this phenomenon is yet to be developed. On the other hand, overparameterization in the context of neural networks is well studied. For e.g., the phenomenon of double-descent has been well studied for overparameterized neural networks, but not explored in PCs. Thus, borrowing such notions to understand the characteristics of PC loss landscapes and building more efficient optimizers that can make use of the tractability of PCs presents a promising avenue for future research.

Latent Representation Learning. Learning semantically meaningful and disentangled latent representations is a fundamental goal in generative modeling. The sum nodes in a PC introduces probabilistically meaningful latent variables. However, learning useful data respresentations using them is non trivial and less explored. Indeed, the works [Vergari *et al.*, 2018; Vergari *et al.*, 2019] have laid some of the foundations in this direction by extracting interpretable representations from a PC by looking at node activations. However a scalable and differentiable approach to learning such representations is still lacking. The recent works [Liu *et al.*, 2023a; Liu *et al.*, 2023b] have enabled distilling information from VAEs within PCs. Extending this framework to support representation learning for PCs is a promising future direction.

Adversarial Training. Maximum likelihood learning employed for training PCs, though stable, is known to achieve suboptimal sample quality. Prior works in the context of DGMs have thus explored augmenting the MLE objective with adversarial losses to improve sample generation. While recent works [Peddi *et al.*, 2022] have studied the robustness of PCs to adversarial attacks, utilizing adversarial losses is relatively less explored. A key challenge here is that sampling in PCs is typically non-differentiable. Hence, unlike GANs, backpropagating the output of an adversarial discriminator for the generated samples is difficult. However, recent works [Shao *et al.*, 2022; Lang *et al.*, 2022] have explored differentiable sampling strategies for PCs, making adversarial

training of PCs a potential way to improve their expressivity.

Incorporating Symmetries. Enabling the adoption of hybrid probabilistic models in real world applications demand that we instill in them the ability to model domain specific inductive biases. While such designs have been explored for domains such as images [Butz *et al.*, 2019], time series [Yu *et al.*, 2022b], and tree structured graphs [Papez *et al.*, 2024], extending them to capture the symmetries, invariances, and equivariances needed for domains involving relational data, sets, and general graphs is an area of active research.

Multi-Modal Learning. With the increasing abundance of heterogenous data, building PCs that can utilize multiple (if not all) modalities of available data to make effective and reliable decisions is an important and open research problem. Recent works on integrating flows with circuits [Sidheekh *et al.*, 2023] have laid the foundations for modeling flexible leaf distributions within PCs. One way to enable probabilistic multi-modal and perhaps compositional learning would thus be to embed normalizing flows trained over different modalities as leaf distributions in PCs.

Applications in Other Fields. Finally, expressive probabilistic models can be used as replacements for DGMs in various learning paradigms. For instance, one of the applications of DGMs within reinforcement learning involves their use as world models [Ha and Schmidhuber, 2018], which enables an agent to hallucinate the behavior of its environment to take better actions. Employing PCs in such a context could additionally empower the agent to perform probabilistic inference over the environment dynamics. Another example is *active feature acquisition*, where DGMs have been employed to assess potential information gain associated with acquiring new features [Li and Oliva, 2021]. A PC, when employed in this context, brings with it the power to exactly and efficiently compute information theoretic quantities of interest and reason over relevant subsets of features via marginalization.

6 Conclusion

We presented an extensive overview of the current research on tractable probabilistic models, focusing on PCs and discussing the various algorithmic and design extensions aimed at improving their expressivity. We also outlined how recent works have attempted to bridge the gap between the expressivity of DGMs and the tractability of PCs by building hybrid models. An important point to highlight is that *structure* is an inevitable factor when learning probabilistic generative models. Stabilizing the training of DGMs such as GANs and VAEs often requires imposing a weak structure over their parameters. The invertible structure imposed by normalizing flows enables exact density evaluation and stable maximum likelihood training. Imposing more restrictive classes of structure over PCs helps us gain tractability over increasingly complex queries. The key focus thus should be to build generative models that can exploit the right level of structure required for solving the tasks at hand. By understanding the design principles of generative modeling and merging them to build hybrid models, we gain the ability to interpolate on this tractability-expressivity spectrum. The possibilities that await such flexible probabilistic models are virtually infinite.

References

[Adel *et al.*, 2015] Tameem Adel, David Balduzzi, and Ali Ghodsi. Learning the structure of sum-product networks via an svd-based algorithm. In *UAI*, 2015.

[Ahmed *et al.*, 2022] Kareem Ahmed, Stefano Teso, Kai-Wei Chang, Guy Van den Broeck, and Antonio Vergari. Semantic probabilistic layers for neuro-symbolic learning. *NeurIPS*, 2022.

[Bond-Taylor *et al.*, 2022] S. Bond-Taylor, A. Leach, Y. Long, and C. G. Willcocks. Deep generative modelling: A comparative review of vaes, gans, normalizing flows, energy-based and autoregressive models. *TPAMI*, 2022.

[Butz *et al.*, 2019] Cory J. Butz, Jhonatan de S. Oliveira, André E. dos Santos, and André L. Teixeira. Deep convolutional sum-product networks. In *AAAI*, 2019.

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

[Choi *et al.*, 2022] YooJung Choi, Tal Friedman, and Guy Van den Broeck. Solving marginal MAP exactly by probabilistic circuit transformations. In *AISTATS*, 2022.

[Correia *et al.*, 2023] Alvaro H. C. Correia, Gennaro Gala, Erik Quaeghebeur, Cassio de Campos, and Robert Peharz. Continuous mixtures of tractable probabilistic models. In *AAAI*, 2023.

[Dang *et al.*, 2020] Meihua Dang, Antonio Vergari, and Guy Broeck. Strudel: Learning structured-decomposable probabilistic circuits. In *PGM*, 2020.

[Dang *et al.*, 2022] Meihua Dang, Anji Liu, and Guy Van den Broeck. Sparse probabilistic circuits via pruning and growing. In *NeurIPS*, 2022.

[Darwiche, 2003] Adnan Darwiche. A differential approach to inference in bayesian networks. *J. ACM*, (3), 2003.

[Dempster *et al.*, 1977] 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)*, 1977.

[Dennis and Ventura, 2012] Aaron W. Dennis and Dan Ventura. Learning the architecture of sum-product networks using clustering on variables. In *NeurIPS*, 2012.

[Di Mauro *et al.*, 2017] Nicola Di Mauro, Antonio Vergari, Teresa MA Basile, and Floriana Esposito. Fast and accurate density estimation with extremely randomized cutset networks. In *ECML PKDD*, 2017.

[Friedman and Koller, 2003] Nir Friedman and Daphne Koller. Being bayesian about network structure. a bayesian approach to structure discovery in bayesian networks. *Machine learning*, 2003.

[Friedman, 1998] Nir Friedman. The bayesian structural em algorithm. In *UAI*, 1998.

[Gala *et al.*, 2024] Gennaro Gala, Cassio de Campos, Robert Peharz, Antonio Vergari, and Erik Quaeghebeur. Probabilistic integral circuits. In *AISTATS*, 2024.

[Gens and Domingos, 2012] Robert Gens and Pedro M. Domingos. Discriminative learning of sum-product networks. In *NeurIPS*, 2012.

[Gens and Domingos, 2013] Robert Gens and Pedro M. Domingos. Learning the structure of sum-product networks. In *ICML*, 2013.

[Ha and Schmidhuber, 2018] David Ha and Jürgen Schmidhuber. Recurrent world models facilitate policy evolution. In *NeurIPS*, 2018.

[He *et al.*, 2016] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

[He *et al.*, 2022] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross B. Girshick. Masked autoencoders are scalable vision learners. In *CVPR*, 2022.

[Heckerman *et al.*, 1995] David Heckerman, Dan Geiger, and David M Chickering. Learning bayesian networks: The combination of knowledge and statistical data. *Machine learning*, 1995.

[Karanam *et al.*, 2024] Athresh Karanam, Saurabh Mathur, Sahil Sidheekh, and Sriraam Natarajan. A unified framework for human-allied learning of probabilistic circuits. *arXiv preprint*, 2024.

[Khosravi *et al.*, 2019] Pasha Khosravi, YooJung Choi, Yitao Liang, Antonio Vergari, and Guy Van den Broeck. On tractable computation of expected predictions. In *NeurIPS*, 2019.

[Kisa *et al.*, 2014] Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *KR*, 2014.

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

[Lang *et al.*, 2022] Steven Lang, Martin Mundt, Fabrizio Ventola, Robert Peharz, and Kristian Kersting. Elevating perceptual sample quality in pcs through differentiable sampling. In *NeurIPS 2021 Workshop on Pre-registration in Machine Learning*, 2022.

[Li and Oliva, 2021] Yang Li and Junier Oliva. Active feature acquisition with generative surrogate models. In Marina Meila and Tong Zhang, editors, *ICML*, 2021.

[Lindsay, 1995] Bruce G Lindsay. Mixture models: theory, geometry, and applications. Ims, 1995.

[Liu *et al.*, 2022] Anji Liu, Stephan Mandt, and Guy Van den Broeck. Lossless compression with probabilistic circuits. In *ICLR*, 2022.

[Liu *et al.*, 2023a] Anji Liu, Honghua Zhang, and Guy Van den Broeck. Scaling up probabilistic circuits by latent variable distillation. In *ICLR*, 2023.

[Liu *et al.*, 2023b] Xuejie Liu, Anji Liu, Guy Van den Broeck, and Yitao Liang. Understanding the distillation process from deep generative models to tractable probabilistic circuits. In *ICML*, 2023.

[Liu *et al.*, 2024] Anji Liu, Mathias Niepert, and Guy Van den Broeck. Image inpainting via tractable steering of diffusion models. In *ICLR*, 2024.

[Loconte *et al.*, 2024] Lorenzo Loconte, Aleksanteri Mikulus Sladek, Stefan Mengel, Martin Trapp, Arno Solin, Nicolas Gillis, and Antonio Vergari. Subtractive mixture models via squaring: Representation and learning. In *ICLR*, 2024.

[Mari *et al.*, 2023] Antonio Mari, Gennaro Vessio, and Antonio Vergari. Unifying and understanding overparameterized circuit representations via low-rank tensor decompositions. In *The 6th Workshop on Tractable Probabilistic Modeling*, 2023.

[Marinescu and Dechter, 2005] Radu Marinescu and Rina Dechter. And/or branch-and-bound for graphical models. In *IJCAI*, 2005.

[Mathur *et al.*, 2023] Saurabh Mathur, Vibhav Gogate, and Sriraam Natarajan. Knowledge intensive learning of cutset networks. In *UAI*, 2023.

[Mauá *et al.*, 2017] Denis D Mauá, Fabio G Cozman, Diarmaid Conaty, and Cassio P Campos. Credal sum-product networks. In *ISIPTA*, 2017.

[Mauá *et al.*, 2018] Denis Deratani Mauá, Diarmaid Conaty, Fabio Gagliardi Cozman, Katja Poppenhaeger, and Cassio Polpo de Campos. Robustifying sum-product networks. *IJAR*, 2018.

[Mauro *et al.*, 2021] Nicola Di Mauro, Gennaro Gala, Marco Iannotta, and Teresa M. A. Basile. Random probabilistic circuits. In *UAI*, 2021.

[Molina *et al.*, 2018] Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting. Mixed sum-product networks: A deep architecture for hybrid domains. In *AAAI*, 2018.

[Papez *et al.*, 2024] Milan Papez, Martin Rektoris, Vaclav Smidl, and Tomáš Pevný. Sum-product-set networks: Deep tractable models for tree-structured graphs. In *ICLR*, 2024.

[Peddi *et al.*, 2022] Rohith Peddi, Tahrima Rahman, and Vibhav Gogate. Robust learning of tractable probabilistic models. In *UAI*, 2022.

[Peharz *et al.*, 2013] Robert Peharz, Bernhard C Geiger, and Franz Pernkopf. Greedy part-wise learning of sum-product networks. In *ECML PKDD*, 2013.

[Peharz *et al.*, 2014] Robert Peharz, Robert Gens, and Pedro Domingos. Learning selective sum-product networks. In *LTPM workshop, ICML*, 2014.

[Peharz *et al.*, 2016] Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *TPAMI*, 2016.

[Peharz *et al.*, 2020a] Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *ICML*, 2020.

[Peharz *et al.*, 2020b] Robert Peharz, Antonio Vergari, Karl Stelzner, Alejandro Molina, Xiaoting Shao, Martin Trapp, Kristian Kersting, and Zoubin Ghahramani. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *UAI*, 2020.

[Pevný *et al.*, 2020] Tomáš Pevný, Václav Smídl, Martin Trapp, Ondřej Poláček, and Tomáš Oberhuber. Sum-product-transform networks: Exploiting symmetries using invertible transformations. In *PGM*, 2020.

[Poon and Domingos, 2011] Hoifung Poon and Pedro M. Domingos. Sum-product networks: A new deep architecture. In *UAI*, 2011.

[Rahman *et al.*, 2014] Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *ECML PKDD*, 2014.

[Rooshenas and Lowd, 2014] Amirmohammad Rooshenas and Daniel Lowd. Learning sum-product networks with direct and indirect variable interactions. In *ICML*, 2014.

[Russell, 2010] Stuart J Russell. *Artificial intelligence a modern approach*. 2010.

[Sánchez-Cauce *et al.*, 2021] Raquel Sánchez-Cauce, Iago París, and Francisco Javier Díez. Sum-product networks: A survey. *TPAMI*, 2021.

[Selvam *et al.*, 2023] Nikil Roashan Selvam, Guy Van den Broeck, and YooJung Choi. Certifying fairness of probabilistic circuits. In *AAAI*, 2023.

[Shao *et al.*, 2022] Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. Conditional sum-product networks: Modular probabilistic circuits via gate functions. *IJAR*, 2022.

[Sharir and Shashua, 2018] Or Sharir and Amnon Shashua. Sum-product-quotient networks. In *AISTATS*, 2018.

[Shih and Ermon, 2020] Andy Shih and Stefano Ermon. Probabilistic circuits for variational inference in discrete graphical models. In *NeurIPS*, 2020.

[Shih *et al.*, 2021] Andy Shih, Dorsa Sadigh, and Stefano Ermon. Hyperspns: Compact and expressive probabilistic circuits. In *NeurIPS*, 2021.

[Sidheekh *et al.*, 2023] Sahil Sidheekh, Kristian Kersting, and Sriraam Natarajan. Probabilistic flow circuits: Towards unified deep models for tractable probabilistic inference. In *UAI*, 2023.

[Tan and Peharz, 2019] Ping Liang Tan and Robert Peharz. Hierarchical decompositional mixtures of variational autoencoders. In Kamalika Chaudhuri and Ruslan Salakhutdinov, editors, *ICML*, 2019.

[Trapp *et al.*, 2019] Martin Trapp, Robert Peharz, Hong Ge, Franz Pernkopf, and Zoubin Ghahramani. Bayesian learning of sum-product networks. In *NeurIPS*, 2019.

[Trapp *et al.*, 2020] Martin Trapp, Robert Peharz, Franz Pernkopf, and Carl Edward Rasmussen. Deep structured mixtures of gaussian processes. In *AISTATS*, 2020.

[Ventola *et al.*, 2020] Fabrizio Ventola, Karl Stelzner, Alejandro Molina, and Kristian Kersting. Residual sum-product networks. In *PGM*, 2020.

[Ventola *et al.*, 2023] Fabrizio Ventola, Steven Braun, Zhongjie Yu, Martin Mundt, and Kristian Kersting. Probabilistic circuits that know what they don't know. In *UAI*, 2023.

[Vergari *et al.*, 2018] Antonio Vergari, Robert Peharz, Nicola Di Mauro, Alejandro Molina, Kristian Kersting, and Floriana Esposito. Sum-product autoencoding: Encoding and decoding representations using sum-product networks. In *AAAI*, 2018.

[Vergari *et al.*, 2019] Antonio Vergari, Nicola Di Mauro, and Floriana Esposito. Visualizing and understanding sum-product networks. *Machine Learning*, 2019.

[Vergari *et al.*, 2021] Antonio Vergari, YooJung Choi, Anji Liu, Stefano Teso, and Guy Van den Broeck. A compositional atlas of tractable circuit operations for probabilistic inference. In *NeurIPS*, 2021.

[Wang and Kwiatkowska, 2023] Benjie Wang and Marta Kwiatkowska. Compositional probabilistic and causal inference using tractable circuit models. In *AISTATS*, 2023.

[Yang *et al.*, 2023] Yang Yang, Gennaro Gala, and Robert Peharz. Bayesian structure scores for probabilistic circuits. In *AISTATS*, 2023.

[Yu *et al.*, 2021] Zhongjie Yu, Mingye Zhu, Martin Trapp, Arseny Skryagin, and Kristian Kersting. Leveraging probabilistic circuits for nonparametric multi-output regression. In *UAI*, 2021.

[Yu *et al.*, 2022a] Zhongjie Yu, Devendra Singh Dhami, and Kristian Kersting. Sum-product-attention networks: Leveraging self-attention in energy-based probabilistic circuits. In *The 5th Workshop on Tractable Probabilistic Modeling*, 2022.

[Yu *et al.*, 2022b] Zhongjie Yu, Fabrizio Ventola, Nils Thoma, Devendra Singh Dhami, Martin Mundt, and Kristian Kersting. Predictive whittle networks for time series. In *UAI*, 2022.

[Yu *et al.*, 2024] Zhongjie Yu, Martin Trapp, and Kristian Kersting. Characteristic circuits. In *NeurIPS*, 2024.

[Zhang *et al.*, 2023] Honghua Zhang, Meihua Dang, Nanyun Peng, and Guy Van den Broeck. Tractable control for autoregressive language generation. In *ICML*, 2023.

[Zhao *et al.*, 2016] Han Zhao, Pascal Poupart, and Geoffrey J Gordon. A unified approach for learning the parameters of sum-product networks. *NeurIPS*, 2016.

[Zuidberg Dos Martires, 2024] Pedro Zuidberg Dos Martires. Probabilistic neural circuits. In *AAAI*, 2024.