Tractable Sharpness-Aware Learning of Probabilistic Circuits

Hrithik Suresh[*1], **Sahil Sidheekh**[*2], **Vishnu Shreeram M.P**[1], **Sriraam Natarajan**[2], **Narayanan Chatapuram Krishnan**[1]

[1] Mehta Family School of Data Science and Artificial Intelligence, Department of Data Science, Indian Institute of Technology Palakkad, Kerala, India

[2] Erik Jonsson School of Engineering & Computer Science, The University of Texas at Dallas, Richardson, TX, USA

Abstract

Probabilistic Circuits (PCs) are a class of generative models that allow exact and tractable inference for a wide range of queries. While recent developments have enabled the learning of deep and expressive PCs, this increased capacity can often lead to overfitting, especially when data is limited. We analyze PC overfitting from a log-likelihood-landscape perspective and show that it is often caused by convergence to sharp optima that generalize poorly. Inspired by sharpness aware minimization in neural networks, we propose a Hessian-based regularizer for training PCs. As a key contribution, we show that the trace of the Hessian of the log-likelihood–a sharpness proxy that is typically intractable in deep neural networks–can be computed efficiently for PCs. Minimizing this Hessian trace induces a gradient-norm-based regularizer that yields simple closed-form parameter updates for EM, and integrates seamlessly with gradient based learning methods. Experiments on synthetic and real-world datasets demonstrate that our method consistently guides PCs toward flatter minima, improving generalization performance.

Introduction

Probabilistic generative models are fundamental to modern machine learning, offering a principled framework for reasoning under uncertainty by modeling data as samples from an unknown underlying distribution. While deep generative models—such as GANs (Goodfellow et al. 2014), VAEs (Kingma and Welling 2014), and Normalizing Flows (Papamakarios et al. 2021)—have excelled in generating high fidelity samples, they sacrifice the ability to do exact inference tractably. This limits their usefulness when downstream tasks require calibrated probabilities. In contrast, Probabilistic Circuits (PCs) (Choi, Vergari, and den Broeck 2020) have emerged as a unifying framework that imposes structural constraints to guarantee efficient and exact inference for a rich set of queries (Vergari et al. 2021), while retaining enough expressivity for real-world applications such as constrained generation (Zhang et al. 2023), image inpainting (Liu, Niepert, and den Broeck 2024), multi-modal fusion (Sidheekh et al. 2025), and Neurosymbolic-AI (Ahmed et al. 2022; Karanam et al. 2025).

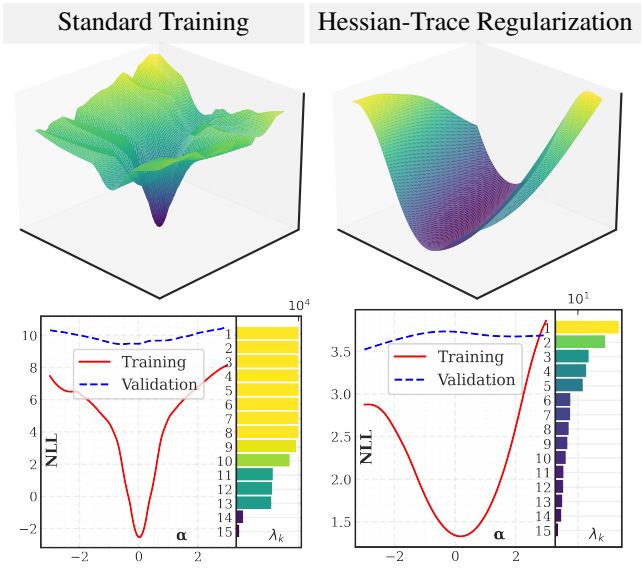

Figure 1: Visualization of the 2D (top) and 1D (bottom) loss-landscape (NLL) near the converged parameters of a PC trained with (right) and without (left) our Hessian trace regularizer on a 2D dataset. Standard training falls into a narrow, sharp basin, while the regularized model settles in flatter minima that generalizes better. The bar plots in the bottom figures depict the top-15 eigenvalues of the hessian at the converged point. The lower eigen spectrum on the right quantifies the reduced sharpness achieved by our method.

Recent works have therefore pushed towards building deeper and more expressive PCs (Sidheekh and Natarajan 2024), with millions of parameters that can be parallelized on GPUs for fast and efficient training/inference (Zhang et al. 2025; Peharz et al. 2020). However, similar to neural networks, deeper and more expressive PC architectures are increasingly prone to overfitting, especially when trained on limited/noisy data. Standard parameter-learning methods can often converge to sharp local optima, leading to poor generalization. Such sharp minima, characterized by high curvature, have been extensively studied in deep neural networks, leading to the development of sharpness-aware optimization methods (Foret et al. 2021; Kwon et al. 2021) that

[*]These authors contributed equally.

explicitly target flatter minima to enhance generalization.

However, to the best of our knowledge, sharpness-aware learning strategies remain relatively unexplored for PCs. We aim to bridge this gap through our work, by studying the geometry of the PC log-likelihood landscape. Our key insight is *that the structural properties of a PC permit efficient and exact computation of second-order geometric information*. In particular, we show that the trace of the Hessian of the log-likelihood—which serves as a measure of surface curvature and a proxy for sharpness– can be computed efficiently in time linear in the number of parameters and the dataset size. This is in stark contrast to deep neural networks, where such exact Hessian computations are intractable in general.

Leveraging this insight, we introduce a Hessian trace regularizer that integrates easily into both gradient-based and Expectation-Maximization (EM) based training of PCs. Crucially, for EM, we derive the closed-form update rule for the sum node parameters, making our approach scalable and easy to integrate into existing training pipelines. To provide an intuitive picture of what our approach accomplishes, we visualize the loss landscape around the converged parameters of a PC trained with and without the Hessian trace regularizer in Figure 1, using the filter-normalized projection technique of Li et al. (2018). The regularized model settles in a broader and flatter optima compared to standard training, verifying our claim that Hessian trace minimization steers the optimization away from sharp valleys, which in turn delivers stronger generalization. Overall, in this work, we make the following contributions:

1. We derive a **closed form expression for the exact full Hessian** of the log-likelihood for tree-structured PCs and show that it can be computed tractably.
2. For general (DAG-structured) PCs, we establish that although the full Hessian can be intractable, its trace remains exactly computable in time linear in both the number of parameters and dataset size, **providing the first practical curvature measure for large-scale PCs.**
3. We introduce a **novel sharpness-aware regularizer** for learning PCs, derived from this Hessian trace.
4. We show that while directly minimizing the Hessian trace via EM leads to a cubic update equation, we can reformulate this objective into an equivalent gradient norm minimization problem, **resulting in a quadratic equation with closed-form parameter updates**.
5. We conduct exhaustive experiments on multiple synthetic and real-world datasets to **show that our regularizer enforces convergence to flatter optima and helps reduce overfitting**, especially in limited data settings.

Background and Preliminaries

Definition 1. A **Probabilistic Circuit** p is a parameterized directed acyclic graph (DAG) with a unique root node n_r that compactly encodes a joint probability distribution over a set of random variables $\mathbf{X} = \{X_1, \ldots, X_d\}$. It is composed of three types of nodes: *Input nodes (leaf)* representing simple univariate distributions over a single variable, *Sum nodes (internal)* that compute a weighted sum of its children's output, and *Product nodes (internal)* that compute a product of its children's output. Formally, each node n in the DAG computes a distribution p_n, defined recursively as follows:

$$p_n(x) = \begin{cases} f_n(x), & \text{if } n \text{ is an input node} \\ \prod_{c \in in(n)} p_c(x), & \text{if } n \text{ is a product node} \\ \sum_{c \in in(n)} \theta_{nc} \cdot p_c(x), & \text{if } n \text{ is a sum node} \end{cases}$$

where f_n is a univariate input distribution (e.g., Bernoulli, Gaussian, etc), $in(n)$ denotes the children of n and θ_{nc} is the weight parameter on the edge (n, c), such that $\forall c \in in(n)\ \theta_{nc} \in (0, 1]$ and $\sum_{c \in in(n)} \theta_{nc} = 1$.

The sum and product nodes represent convex mixtures and factorized distributions over the scopes of their children, respectively. A PC is evaluated bottom up and the joint distribution is computed as the output of its root node, i.e. $p(x) = p_{n_r}(x)$. The size of p, denoted $|p|$, is the number of edges in its DAG. We make the common assumption that p contains alternating sum and product node layers. This formalism subsumes several classes of tractable models such as arithmetic circuits (Darwiche 2003), sum-product networks (Poon and Domingos 2011), PSDDs (Kisa et al. 2014) and cutset networks (Rahman, Kothalkar, and Gogate 2014).

To achieve tractability for exact marginal (MAR), conditional (CON) and maximum-a-posteriori (MAP) inference, a PC has to satisfy certain **structural properties** (Choi, Vergari, and den Broeck 2020), such as *smoothness*, which ensures that each sum node represents a valid mixture, and *decomposability*, which allows integrals (or sums) to factorize recursively for tractable MAR and CON inference. However enforcing structural properties often reduces the model's expressivity. Thus, recent works have aimed to increase their expressivity by efficiently scaling them using tensorized implementations (Peharz et al. 2019, 2020; Liu, Ahmed, and den Broeck 2024; Loconte et al. 2025), borrowing inductive biases from deep generative models (Sidheekh, Kersting, and Natarajan 2023; Liu, Zhang, and den Broeck 2023; Correia et al. 2023; Gala et al. 2024), and relaxing assumptions on the parameters (Loconte et al. 2024; Loconte, Mengel, and Vergari 2025; Wang and Van den Broeck 2025) as well as structure of the PC (Sharir and Shashua 2018; Shao et al. 2022; Dos Martires 2024).

Regardless of these advances, learning the PC parameters is predominantly achieved using one of two standard paradigms: stochastic gradient based optimization or expectation-maximization (EM). As differentiable computational graphs, PCs allow efficient computation of gradients of the likelihood (or log-likelihood) w.r.t their parameters via backpropagation. Thus stochastic gradient descent (SGD) and its variants can be used directly to learn their parameters. Alternatively, one can view each sum node as introducing a latent categorical variable indexing its child edges and apply EM to optimize the incomplete data log-likelihood. In this framework, the E-step computes posterior distributions over edges, while the M-step updates the weights in closed form, given the posterior. Notably, both SGD and EM admit a unified formulation that can be understood in terms of the notion of *circuit flow* (Liu and den Broeck 2021).

Definition 2 (Circuit Flow)**.** The flow associated with the

nodes of a PC is defined in the following recursive manner

$$F_n(x) = \begin{cases} 1, & \text{if } n \text{ is the root} \\ \sum_{m \in \mathrm{pa}(n)} F_m(x), & \text{if } n \text{ is a sum/input} \\ \sum_{m \in \mathrm{pa}(n)} \theta_{mn} \frac{p_n(x)}{p_m(x)} F_m(x), & \text{if } n \text{ is a product} \end{cases}$$

The flow associated with an edge (n, c) is defined as $F_{nc}(x) = \theta_{nc} \frac{p_c(x)}{p_n(x)} F_n(x)$. In the EM interpretation, $F_{nc}(x)$ corresponds to the expected count of how often the edge (nc) is used (E-step). The M-step then maximizes $\sum_{c \in \mathrm{in}(n)} F_{nc}(x) \log \theta_{nc}$ s.t. $\sum_{c \in \mathrm{in}(n)} \theta_{nc} = 1$, which for a mini-batch D_i yields the closed form update $\theta_{nc}^{\text{mini}} = \frac{\sum_{x \in D^i} F_{nc}(x)}{\sum_{j \in \mathrm{in}(n)} \sum_{x \in D^i} F_{nj}(x)}$. To smooth out mini-batch noise, a running average is often used: $\theta_{nc}^{\text{new}} = (1-\alpha)\, \theta_{nc}^{\text{old}} + \alpha\, \theta_{nc}^{\text{mini}}, \quad \alpha \in [0, 1]$. Under gradient-based learning, flows have the interpretation that $F_{nc}(x) = \theta_{nc} \frac{\partial \log P_{n_r}(x)}{\partial \theta_{nc}}$. Computing the flows requires only a single forward-backward pass of the PC, and has been proven effective for learning PCs with hundreds of millions of parameters at scale (Liu, Ahmed, and den Broeck 2024).

However, such large and expressive PCs can overfit when data is limited, prompting **regularization strategies** that adapt ideas from both deep learning and graphical models. Probabilistic dropout (Peharz et al. 2019) randomly masks inputs and sum-node children during training to simulate missing data and mixture uncertainty, reducing co-adaptation among sub-circuits. Pruning and re-growing (Dang, Liu, and den Broeck 2022) has been suggested to remove redundant sub-networks, yielding sparser models that generalize better. Classical Laplacian smoothening on the sum-node weights have also been applied to PCs (Liu and den Broeck 2021), although naive Laplace priors can bias the mixtures when child supports are imbalanced. Shih, Sadigh, and Ermon (2021) observed that deep PCs can have tens of millions of parameters and proposed a hypernetwork that generates sum-node weights from low dimensional embeddings, thereby reducing the free parameters while preserving expressivity. Recent work has also tried to exploit the tractability of PCs to propose customized regularizers. Ventola et al. (2023) adapted the idea of Monte Carlo dropout (Gal and Ghahramani 2016) to PCs by deriving tractable dropout inference that propagates variances through the circuit in a single pass to obtain better calibration and out-of-distribution detection. Liu and den Broeck (2021) proposed *data softening*, which replaces each training example with a locally blurred distribution and an entropy regularizer on the circuit's global output distribution. However, this requires solving a non-linear equation via Newton's method at each sum node, leading to added implementation and computational complexity during training.

In parallel, the deep-learning community has demonstrated that convergence to sharp minima–regions of high curvature–often correlates with poor generalization, leading to the development of **sharpness-aware optimization strategies**. While the idea that flatter minima can help unlock higher generalization capability in deep neural networks (DNNs) has been around for a long time (Hochreiter and Schmidhuber 1997), recent works have shown how to achieve this in practice using sharpness aware minimization (SAM) (Foret et al. 2021), which solves a local minimax problem to find parameter updates robust to worst-case perturbations. Extensions such as Adaptive SAM (ASAM) (Kwon et al. 2021) have also been proposed to adaptively scale the perturbations using curvature information. We **posit that the relevance of sharpness-aware techniques extends beyond DNNs to PCs**, where training objectives like log-likelihood maximization are often non-convex and prone to overfitting in overparameterized regimes. In such settings, sharpness-aware learning can not only offer a principled means to promote solutions that generalize better, but the structure inherent in PCs can enable computing such curvature information *exactly* and *efficiently*.

Sharpness-Aware Learning for PCs

The Hessian matrix-the second order partial derivatives of a loss function-has long been used as a natural way to quantify flatness, as its eigenvalues capture the curvature along different directions. Böttcher and Wheeler (2024) used dominant eigenvectors to visualize the loss landscape of DNNs and distinguish sharp minima from flat ones, while Chaudhari et al. (2017) used this idea to guide DNNs towards wider optima. More recently, Kaur, Cohen, and Lipton (2023) proposed using the largest eigenvalue as a flatness metric, while Sankar et al. (2021) developed a layer-wise Hessian trace-based regularizer. However, as computing the full Hessian is intractable for DNNs, most methods rely on implicit curvature estimates, such as rank-1 approximations (Martens, Sutskever, and Swersky 2012) or the Hutchinson trace estimator (Hutchinson 1990). Recent theoretical work by Maene and De Raedt (2025) on algebraic model counting suggests that computing the full Hessian for PCs incurs a quadratic memory cost even for smooth, decomposable, and deterministic circuits. In this work, we study the tractability of the Hessian for PCs in further detail, and show that their structural properties enable exact and efficient Hessian-trace computations, allowing a true sharpness-aware regularizer without resorting to costly approximations typical in DNNs.

Full Hessian Computation for Tree-Structured PCs

A tree-structured PC (in short TS-PC) is one where every non-root node in its DAG has exactly one parent, and hence there is a unique path from the root (n_r) to any node (n). Our first result is that for a TS-PC, the Hessian of the log-likelihood with respect to the parameters can be computed tractably. Figure 2 illustrates a typical n_r–n path in a TS-PC. Let $S_i^l(x)$ and $P_i^l(x)$ denote the outputs of the i^{th} sum and product nodes at level l, respectively. Note that x would contain only the subset of variables defined in the scope of the node. We ignore it in the notation for clarity. To see how this structure yields closed-form Hessian entries, consider the task of expressing the gradient of the root likelihood $P_{n_r}(x)$ w.r.t a mixing weight θ_{nc}. We can unfold the computation

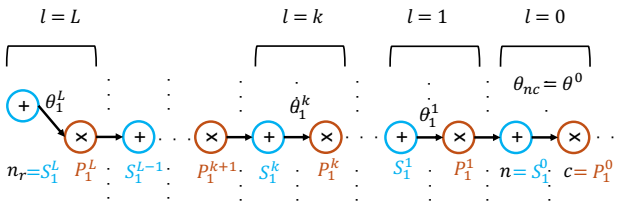

Figure 2: A typical path structure in a tree-structured PC.

along a unique path $n_r \to \cdots \to P_i^l \to S_j^l \to \cdots n \to c$. At each product node P_i^l on this path, the contribution of its other children, i.e. those not on the path enters as a multiplicative factor. We will refer to the product of such *sibling* outputs as the *product complement*, as defined below:

Definition 3 (Product Complement)**.** The product complement of a product node P_i^l with respect to one of its children S_j^{l-1} is defined as: $\overline{P}_{ij}^l(x) = \prod_{\substack{k \in in(P_i^l) \\ k \neq j}} S_k^{l-1}(x)$

All such product complements in a PC can be computed in a single forward pass. Recall that a single forward-backward pass also computes the *circuit flow* $F_n(x)$ (Def. 2) for each node n and $F_{nc}(x)$ for each edge (n, c), which can be used to compute the gradients. The unique-path property of a TS-PC collapses the summations in the circuit flow recursion into a simpler chain, resulting in compact expressions for the flow and gradient, as presented below.

Lemma 1. Consider the unique path from the root n_r to a sum node n in a TS-PC: $n_r \to P_1^L \to S_1^{L-1} \to \cdots \to P_1^1 \to S_1^0 = n$, as shown in Figure 2. Then the flow at node n is given by: $F_n(x) = \theta_1^1 \frac{P_1^1(x)}{P_{n_r}(x)} \prod_{j=2}^{L} \theta_1^j \bar{P}_{11}^j(x)$

Proof. The proof involves unrolling the flow at a sum node using the tree structure of a TS-PC and the notion of product complements of the product nodes along the path from the node to the root, and is provided in the appendix (Suresh et al. 2025). □

Corollary 1. The flow associated with the edge (n, c) parametrized by θ_c^n ($= \theta_1^0$ in Figure 2) is given by: $F_{nc}(x) = \theta_{nc} \frac{P_c(x)}{P_{n_r}(x)} \prod_{l=1}^{L} \theta_1^l \bar{P}_{11}^l(x)$ and correspondingly, the gradient of the likelihood at the root node with respect to θ_{nc} is $\frac{\partial P_{n_r}(x)}{\partial \theta_{nc}} = P_c(x) \prod_{l=1}^{L} \theta_1^l \bar{P}_{11}^l(x)$

Proof. The proof is a straight forward application of the flow defined on an edge. □

For notational simplicity we have labeled every sum and product-node along our canonical path by index "1". In a general TS-PC path, the nodes at layer l would carry their own index i_l, but the same formulas hold by replacing each "1" with the appropriate i_l. The key insight from the above corollary is that the partial derivative of the likelihood with respect to θ_{nc} factorizes into exactly the product complements and weights along the unique path from n to the root, and *does not depend on any sum-node outputs* on that path.

Let θ_{nc} and $\theta_{n'c'}$ be the weights associated with two distinct edges in a TS-PC. Due to the tree-structure, n and n' either lie on the same path from the root node or share exactly one deepest common ancestor, let us call it q. Thus, the pair $(\theta_{nc}, \theta_{n'c'})$ belongs to one of the following three cases: 1) **Sum pair**: if q is a sum node. 2) **Product pair**: if q is a product node. 3) **Path pair**: if the θ_{nc} and $\theta_{n'c'}$ lie on the same path from the root. Our main result shows that the second derivative of root likelihood w.r.t the PC parameters can be expressed in closed form for each of the above cases.

Theorem 1. The mixed second derivative $\frac{\partial^2 P_{nr}(x)}{\partial \theta_{n'c'} \partial \theta_{nc}}$ of the output of a tree-structured PC with respect to any two parameters θ_{nc} and $\theta_{n'c'}$ equals

$$\begin{cases} 0, & \text{if it is a sum pair} \\ \frac{\frac{\partial P_{nr}(x)}{\partial \theta_{nc}} \frac{\partial P_{nr}(x)}{\partial \theta_{n'c'}}}{\theta_1^k P_1^k(x) \left(\prod_{l=k+1}^{L} \theta_1^l \bar{P}_{11}^l(x)\right)}, & \text{if it is a product pair} \\ \frac{1}{\theta_{n'c'}} \cdot \frac{\partial P_{nr}(x)}{\partial \theta_{nc}}, & \text{if it is a path pair with } \theta_{n'c'} \text{ closer to the root} \end{cases}$$

where P_1^k denotes the deepest common product node, and we follow our canonical notation for the path to the root.

Proof. Deferred to the appendix. □

Theorem 1 can be further extended to obtain equally simple expressions for the Hessian of the *log-likelihood* which is typically used as the objective for training PCs, as follows:

Proposition 1. If $(\theta_{nc}, \theta_{n'c'})$ is a sum pair, then

$$\frac{\partial^2 \log P_{n_r}(x)}{\partial \theta_{n'c'} \partial \theta_{nc}} = -\frac{F_{nc}(x)}{\theta_{nc}} \frac{F_{n'c'}(x)}{\theta_{n'c'}} \tag{1}$$

Proof. Deferred to the appendix. □

Thus, the mixed second derivative of the log-likelihood for a sum pair factorizes into the negative product of the corresponding first-order derivatives. *A special case is when the sum pair corresponds to the same parameter ($\theta_{nc} = \theta_{n'c'}$), in which case the double derivative equals the negative square of the gradient of the log-likelihood with respect to θ_{nc}.*

Proposition 2. If $(\theta_{nc}, \theta_{n'c'})$ is a product pair with P_1^k being their deepest common ancestor, then

$$\frac{\partial^2 log P_{n_r}(x)}{\partial \theta_{n'c'} \partial \theta_{nc}} = \frac{P_{n_r}(x) \frac{F_{nc}(x)}{\theta_{nc}} \frac{F_{n'c'}(x)}{\theta_{n'c'}}}{\theta_1^k P_1^k(x) \left(\prod_{l=k+1}^{L} \theta_1^l \bar{P}_{11}^l(x)\right)} - \frac{F_{nc}(x)}{\theta_{nc}} \frac{F_{n'c'}(x)}{\theta_{n'c'}}$$

Proof. Deferred to the appendix. □

Proposition 3. If $(\theta_{nc}, \theta_{n'c'})$ is a path pair, with $\theta_{n'c'}$ closer to the root node, then

$$\begin{aligned} \frac{\partial^2 \log P_{n_r}(x)}{\partial \theta_{n'c'} \partial \theta_{nc}} &= \frac{\partial^2 \log P_{n_r}(x)}{\partial \theta_{nc} \partial \theta_{n'c'}} \\ &= \frac{1}{\theta_{n'c'}} \cdot \frac{F_{nc}(x)}{\theta_{nc}} - \frac{F_{nc}(x)}{\theta_{nc}} \frac{F_{n'c'}(x)}{\theta_{n'c'}} \end{aligned}$$

Proof. Deferred to the appendix. □

Since each entry of the full Hessian depends only on the circuit flow, the mixing parameters and the product complements–all of which can be computed efficiently–**the Hessian as a whole can likewise be evaluated tractably**.

Hessian for General (DAG-Structured) PCs

To understand whether the tractability of Hessian computation extends to general DAG-structured PCs, we examine its diagonal and off-diagonal entries separately. Our analysis suggests that while the former can be computed efficiently, the latter can suffer from a combinatorial explosion. We present the results and defer proofs to the supplementary.

Proposition 4. The diagonal entry of the Hessian of the log-likelihood of a general PC w.r.t a parameter θ_{nc} is given by: $\frac{\partial^2 \log P_{n_r}(x)}{\partial^2 \theta_{nc}} = -\left(\frac{F_{nc}(x)}{\theta_{nc}}\right)^2$

From Proposition 4, we observe that computing the trace only requires access to the edge flows and the mixing parameters. From Definition 2, all edge flows can be computed with a single forward and backward pass through the circuit, making the flow computation linear in time with respect to the number of edges. Consequently, the overall trace computation is also linear in time. However, for general PCs, we conjecture that computing the off-diagonal entries of the Hessian is intractable due to the exponential number of dependency paths between parameters. Concretely, consider a PC where each internal node (except the root and its immediate children) has up to w parents. Suppose that sum nodes n and n' share w deepest common ancestors at the same depth, and let d^* denote the number of layers between these deepest common ancestors and the lower of the two nodes, i.e., $d^* = \min(\text{depth}(n), \text{depth}(n')) - \text{depth}(\text{deepest common ancestors})$. Then, the number of paths connecting n and n' can grow as $O(w^{d^*})$. This exponential growth in the number of shared paths indicates that, in such densely connected PCs, computing off-diagonal entries of the Hessian becomes computationally intractable.

A Tractable Sharpness Regularizer for PCs

Although the full Hessian can be intractable for arbitrary PCs, its trace remains efficiently computable, and serves as a scalar measure of the overall curvature of the log-likelihood surface–large values indicating sharper optima, while lower values correspond to flatter optima (Keskar et al. 2017; Foret et al. 2021; Kwon et al. 2021). Thus, reducing the Hessian trace during training can serve as an effective regularization strategy. While the full Hessian can be computed efficiently and potentially incorporated as a regularizer for tree-structured PCs, we focus instead on its trace, as it is both simpler to compute and applicable to the general class of PCs. Crucially, for any PC (not just tree-structured), the absolute trace (ignoring *absolute* henceforth) simplifies to the sum of squared partial derivatives: $\text{Tr}\left(\nabla^2 \log P_{n_r}(x)\right) = \sum_{n,c}\left(\frac{\partial \log P_{n_r}(x)}{\partial \theta_{nc}}\right)^2 = \sum_{n,c}\left(\frac{F_{nc}(x)}{\theta_{nc}}\right)^2$ This enables sharpness-aware regularization using only first-order derivatives that can be computed using the edge flows, while still promoting flatter solutions during training. A simple way to incorporate this into gradient-based learning is to add the Hessian trace as a regularizer $R(\theta, x) = \sum_{n,c}(F_{nc}(x)/\theta_{nc})^2$ to the negative log-likelihood, yielding the objective $\min_\theta \; -\sum_{x\in D} \log P_{n_r}(x) \; + \; \lambda \sum_{x\in D} R(\theta, x)$. Since $R(\theta, x)$ depends only on the local flows and weights, its gradients can be computed with a forward-backward pass, making integration with optimizers like SGD or Adam straightforward. However, EM is often preferred over gradient descent to learn PCs as it achieves faster convergence (Desana and Schnörr 2017). Thus, we next discuss how to integrate this curvature penalty into EM-based learning.

Sharpness-Aware EM. To endow EM with sharpness awareness, we propose to add the Hessian-trace regularizer into the M-step by constraining the sum squared gradients at each sum node. More formally, the M-step optimization is now carried out under two constraints: (1) the parameters at each sum node must lie on the probability simplex, i.e., $\sum_{c\in in(n)} \theta_{nc} = 1$, and (2) the trace of the Hessian is upper bounded, i.e., $\sum_{c\in in(n)} \left(F_{nc}(x)/\theta_{nc}\right)^2 \leq m$ for some m. The resulting parameter update takes the following form:

Proposition 5. The EM update for a parameter θ_{nc} at a sum node n, under a Hessian trace-based sharpness regularizer, is the solution to the cubic equation:

$$\lambda\, \theta_{nc}^3 - F_{nc}(x)\, \theta_{nc}^2 - 2\,\mu F_{nc}(x)^2 = 0, \tag{2}$$

where $F_{nc}(x)$ is the expected edge flow, and λ, μ are Lagrange multipliers for the normalization and trace constraints, respectively.

Incorporating the Hessian trace into the EM update thus yields, at each sum node, a cubic equation in the parameters that must be solved exactly. However, solving such cubic equations can be computationally expensive and may yield multiple real roots, negative values, or even complex solutions, making the update process unstable or infeasible. To overcome this, we exploit a key property of PC gradients: the partial derivative of the log-likelihood with respect to a sum node parameter θ_{nc} takes the form $\frac{F_{nc}(x)}{\theta_{nc}}$, which is always non-negative as both $F_{nc}(x)$ and θ_{nc} are positive. Consequently, the squared gradient is a monotonic function of the gradient itself, thus minimizing the gradient suffices to reduce its square. This allows us to directly penalize the gradient as a surrogate for reducing the trace, resulting in a simpler *quadratic* update with a closed-form solution.

Theorem 2. The EM parameter update at sum node n, under the gradient regularized objective is given by:

$$\theta_{nc} = \frac{F_{nc}(x) + \sqrt{F_{nc}(x)^2 + 4\lambda\mu F_{nc}(x)}}{2\lambda},$$

where $F_{nc}(x)$ denotes the flow along the edge from sum node n to its child c, and $\lambda, \mu \geq 0$ are the Lagrange multipliers corresponding to the normalization and regularization constraints, respectively.

Experiments and Results

We organize our empirical evaluation to answer the following four research questions.

(**Q1**) Do large, expressive PCs overfit on limited data, and does sharpness, defined via Hessian-trace capture this?

(**Q2**) Is our derived sum-squared-gradient expression for the Hessian-trace efficient to compute?

Dataset	1%			5%			10%			50%			100%		
	ΔNLL	ΔDoF	ΔSharp	ΔNLL	ΔDoF	ΔSharp	ΔNLL	ΔDoF	ΔSharp	ΔNLL	ΔDoF	ΔSharp	ΔNLL	ΔDoF	ΔSharp
bent_lissajous	46.65	65.62	93.41	19.45	28.23	52.87	18.16	22.21	56.44	1.27	2.01	34.58	0.65	−0.06	31.25
helix	32.38	62.87	91.80	17.92	35.29	57.51	10.85	5.48	21.48	8.93	−1.90	13.76	7.33	−1.76	15.71
interlocked_circles	41.19	68.16	88.06	26.54	39.14	75.72	18.65	14.55	46.30	2.20	1.14	34.15	1.02	0.40	41.59
knotted	52.66	59.88	91.01	26.68	42.31	65.46	17.06	9.42	31.12	5.67	1.23	25.44	10.07	−0.01	21.57
pinwheel	57.29	61.85	93.85	22.62	23.27	66.14	13.24	17.58	75.54	3.70	6.14	76.26	0.38	0.77	79.01
spiral	62.11	69.82	92.23	34.03	37.47	67.09	22.62	20.49	61.48	19.83	1.38	12.66	18.35	3.78	23.72
twisted_eight	49.03	70.20	75.09	13.65	22.31	63.21	11.22	12.54	51.52	2.72	2.42	32.11	−1.22	−0.04	40.85
two_moons	54.92	67.30	88.47	27.99	30.24	56.63	19.72	17.88	34.10	16.22	−0.13	30.68	3.80	0.85	28.31
Mean	49.53	65.71	89.24	23.61	32.28	63.08	16.44	15.02	47.25	7.57	1.54	32.45	5.05	0.49	35.25

Table 1: Percentage improvements in test negative log-likelihood (ΔNLL), reduction in overfitting (ΔDoF), and percentage decrease in the loss-surface sharpness ($\Delta Sharp$) for an **Einsum Network** on the **synthetic manifold datasets**, comparing models with Hessian-trace regularization against vanilla counterparts. Values are averaged across 5 independant runs.

(**Q3**) Does the proposed sharpness aware-learning framework reduce overfitting and improve generalization?

(**Q4**) What effect does the regularization strength μ have?

Setup. To answer these, we conducted experiments on 8 synthetic 2D/3D manifold datasets (Sidheekh, Kersting, and Natarajan 2023) as well as the 20 standard binary density estimation benchmark. To show that our approach applies broadly across different PC model classes and implementations, we integrated it into two widely used PC frameworks—Einsum Networks (Peharz et al. 2020) and PyJuice (Liu, Ahmed, and den Broeck 2024), evaluating it on different structural settings. For synthetic datasets, we use a fixed RAT-SPN (Peharz et al. 2019) architecture with 10 input-distributions, sum-nodes and num-repetitions. For the binary datasets, we adopt the Hidden Chow-Liu Tree (HCLT) structure from PyJuice, with a latent size of 100 to increase model capacity. To simulate limited data settings where overfitting can happen, we train each model on random subsets of each dataset at fractions $\{1\%, 5\%, 10\%, 50\%$ and $100\%\}$. To show applicability across learning methods, we integrated our regularizer into two settings: (1) gradient-based learning for Einsum Networks (using Adam) and (2) EM-based learning for PyJuice HCLTs (using our quadratic closed-form updates). We provide further experimental results and details in the appendix, along with our code repository[1].

(Q1) Overfitting and Sharpness: To show that deep PCs indeed overfit when data is scarce, we plot the train and validation negative log-likelihoods (NLL) for an EinsumNet trained on a 5% data for $2D$ spiral distribution in Figure 3[left]. We see that the train-NLL continues to decrease, while the val-NLL rises, indicating overfitting and a widening generalization gap. Crucially, the value of sharpness, computed via Hessian-trace also grows in tandem with this gap, peaking when overfitting occurs. This confirms that sharp minima that correlate with overfitting do exist in PCs, and that the Hessian-trace is capable of detecting them.

(Q2) Efficiency of Hessian-Trace Computation. We empirically validated the correctness of our Hessian trace derivation using Einsum Networks, which support full Hessian evaluation via PyTorch's automatic differentiation.

[1]https://github.com/starling-lab/sharpness-aware-pc

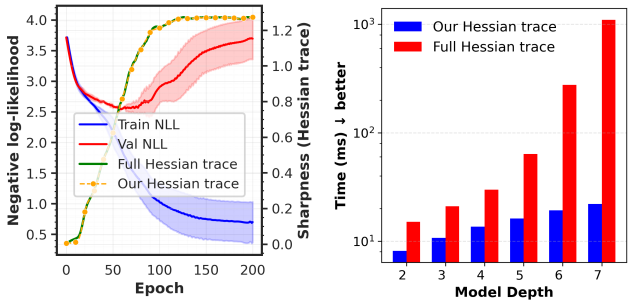

Figure 3: [Left] Training and validation negative log-likelihood of EinsumNet on a $2D$ spiral dataset across epochs, with Hessian-trace computed by our sum-squared-gradients formula and torch autograd. [Right] Time taken for computing the Hessian trace as the network depth grows.

Figure 3[left] shows an exact match between the Hessian trace computed using our proposed sum-of-squared-gradients (SSG) formula and the one obtained directly via autograd on the 2D spiral dataset, where full Hessian computation was feasible without exceeding memory limits. Figure 3[right] also compares the computation time of the Hessian trace using autograd and our closed-form SSG formula. We see that as the model depth increases, autograd's runtime suffers from an exponential blow up, while our approach scales linearly, and is thus a more practical and accurate way to analyze the curvature, even for deep PCs.

(Q3) Gains from Sharpness-Aware Learning. To study the effect our sharpness regularizer has in learning better PCs, we measured the performance improvement achieved by our regularized model as compared to its vanilla counterpart using three metrics-(1) The relative reduction in Test-NLL [$\Delta NLL(\%)$], (2) The reduction in the degree of overfitting [$\Delta DoF(\%)$], where $DoF = \frac{NLL_{test} - NLL_{train}}{|NLL_{train}|}$ and (3) The relative reduction in sharpness [$\Delta Sharp(\%)$], as measured by our Hessian-trace formula, at convergence. Table 1 reports the mean results over five runs for an EinsumNet trained on the synthetic 2D/3D manifold datasets at varying training-set fractions. We observe that in the lowest

Dataset	1%			5%			10%			50%			100%		
	ΔNLL	ΔDoF	ΔSharp	ΔNLL	ΔDoF	ΔSharp	ΔNLL	ΔDoF	ΔSharp	ΔNLL	ΔDoF	ΔSharp	ΔNLL	ΔDoF	ΔSharp
accidents	1.40	6.94	15.61	−1.88	1.54	16.54	−1.67	0.67	12.33	−0.50	0.04	12.03	−0.29	0.01	3.39
ad	2.11	1.24	56.79	3.20	3.54	78.44	1.26	2.65	27.21	−2.26	1.42	10.80	−1.85	0.43	9.31
baudio	9.39	24.87	9.33	−0.58	1.30	3.06	−0.56	0.35	5.17	−0.20	0.00	1.19	−0.11	0.00	0.88
bbc	10.45	3.52	32.12	15.75	19.70	25.16	8.75	19.57	28.21	−0.23	1.32	11.47	−0.37	0.25	10.74
bnetflix	3.07	10.71	−1.65	−0.38	0.29	2.94	−0.31	0.12	2.68	−0.02	0.00	0.30	−0.01	0.00	0.14
book	9.56	6.11	74.56	1.47	4.15	29.80	−0.60	1.09	21.63	−0.83	0.03	10.23	−0.48	0.02	7.63
c20ng	11.56	13.28	31.06	0.78	3.03	25.84	0.06	0.84	10.53	−0.28	0.01	6.30	−0.12	0.01	3.54
cr52	10.54	9.76	35.21	3.68	12.54	−2.54	−0.10	4.04	3.47	−0.57	0.24	3.45	−0.31	0.04	2.65
cwebkb	9.05	4.05	30.95	14.17	26.46	30.52	3.83	12.36	28.79	−0.63	0.39	11.21	−0.46	0.11	5.29
dna	29.45	15.54	3.87	2.59	6.88	13.93	0.10	2.30	8.95	−0.89	0.25	8.81	−0.47	0.07	3.91
jester	19.22	28.28	25.52	0.24	3.81	−2.60	−0.71	0.36	−3.30	−0.24	0.04	0.88	−0.12	0.00	−0.89
kdd	0.04	1.51	11.96	0.11	0.16	3.50	0.10	0.08	7.49	−0.02	0.00	3.65	0.00	0.00	2.20
kosarek	1.93	6.84	23.70	−0.76	0.60	13.33	−0.42	0.27	7.90	−0.20	0.03	7.51	−0.07	0.00	5.66
msnbc	−0.10	0.03	0.35	0.02	−0.01	0.04	−0.02	0.00	0.07	−0.01	0.00	0.06	0.00	0.00	0.06
msweb	1.11	3.75	33.03	−0.23	0.79	18.57	−0.29	0.45	7.33	−0.08	0.04	10.80	−0.03	0.03	6.54
nltcs	0.18	2.31	7.20	−0.94	0.35	4.80	−0.68	0.06	2.52	−0.08	0.00	−1.04	−0.07	0.00	−0.58
plants	1.59	7.19	13.72	−2.41	0.95	10.34	−1.63	0.38	8.02	−0.56	0.02	3.56	−0.28	0.01	3.50
pumsb_star	2.04	4.83	8.62	−2.76	1.09	16.12	−1.69	0.51	8.68	−0.50	0.05	5.50	−0.29	0.02	4.33
tmovie	15.15	15.90	16.97	12.01	28.27	−3.17	0.02	8.32	9.83	−1.42	0.34	3.55	−0.82	0.12	11.08
tretail	4.55	9.60	31.95	−0.25	0.44	21.81	−0.10	0.12	10.93	−0.03	0.00	0.69	−0.02	0.00	1.44
Mean	7.12	8.81	23.04	2.19	5.79	15.32	0.27	2.73	10.42	−0.48	0.21	5.55	−0.31	0.06	4.04

Table 2: Percentage improvements in test negative log-likelihood (ΔNLL), reduction in overfitting (ΔDoF), and percentage decrease in the loss-surface sharpness ($\Delta Sharp$) for a **PyJuice HCLT** model on the **binary density estimation datasets**, comparing models with Hessian-trace regularization against vanilla counterparts. Values represent the mean over 5 runs.

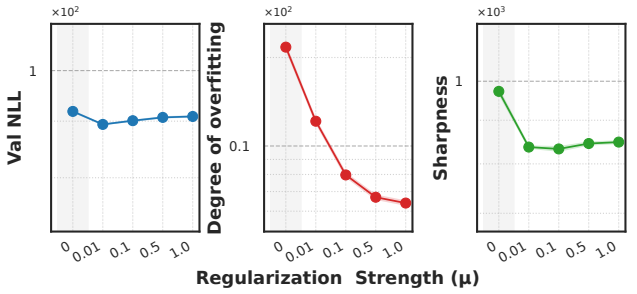

Figure 4: Ablation showing the effect of μ on the validation negative log-likelihood, degree of overfitting and sharpness.

data setting, on average, our regularizer cuts overfitting by up to 65 %, flattens the loss surface by 89%, and boosts test log-likelihood by upto 49%. Although the absolute gains diminish with more data, they remain positive across all fractions on average, demonstrating that trace minimization consistently guides the learning toward better-generalizing optima. Table 2 presents analogous results for PyJuice HCLTs on real-world binary datasets using our regularized EM. Again, in the lowest-data regime we observe a 7 % improvement in test NLL, an 8% reduction in overfitting, and a 23% decrease in sharpness, on average. As dataset size grows, these gains plateau—and at the highest data fractions we record a marginal ($< 0.5\%$) drop in test NLL. This is expected when overfitting is negligible as the regularizer may push the parameters away from an otherwise sufficient optimum. But even in these settings, our method continues to reduce overfitting and sharpness, confirming its effectiveness at steering PCs toward flatter, more robust solutions.

(Q4) Ablation on μ. To study the sensitivity of our framework to the regularization strength, we trained a PyJuice HCLT using the 5% split of DNA binary dataset, for varying values of μ and tracked changes in Validation-NLL, degree of overfitting and sharpness (in Figure 4). We observe that even a small $\mu \in (0, 0.1]$ is sufficient to capture most of the gains-reducing overfitting and curvature. Larger μ values yield only marginal gains at the cost of slight underfitting. Thus, our framework is robust to the choice of μ in a broad mid-range and we select it based on validation performance.

Conclusion

In this work, we introduced a new direction to study the training of PCs through the lens of the log-likelihood surface geometry. We derived a closed-form expression for the exact full Hessian of the log-likelihood for tree-structured PCs. For general DAG-structured PCs, we showed that while the full Hessian can be intractable, its trace remains exactly and efficiently computable– offering the first scalable curvature measure for training large PCs. Building on this, we designed a novel regularizer whose equivalent gradient-norm formulation yields closed-form quadratic updates, enabling efficient optimization. Our experiments confirmed that our approach steers training toward flatter minima and reduces overfitting, especially in low-data regimes. Overall, our work opens up a promising new direction for studying PCs. Future work includes investigating the presence of asymmetric valleys in the loss landscape, analogous to those observed in DNNs, developing a theoretical framework for understanding convergence in over-parameterized PCs, and designing optimization strategies that can better exploit tractable geometric information.

Acknowledgements

SN and SS gratefully acknowledge the generous support by the AFOSR award FA9550-23-1-0239, the ARO award W911NF2010224 and the DARPA Assured Neuro Symbolic Learning and Reasoning (ANSR) award HR001122S0039. CK and HS gratefully acknowledge Anagha Sabu for the discussions related to the work and CK, HS, and VS thank IIT Palakkad for the access to Madhava Cluster.

References

Ahmed, K.; Teso, S.; Chang, K.; den Broeck, G. V.; and Vergari, A. 2022. Semantic Probabilistic Layers for Neuro-Symbolic Learning. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022*.

Böttcher, L.; and Wheeler, G. 2024. Visualizing high-dimensional loss landscapes with Hessian directions. *Journal of Statistical Mechanics: Theory and Experiment*.

Chaudhari, P.; Choromanska, A.; Soatto, S.; LeCun, Y.; Baldassi, C.; Borgs, C.; Chayes, J. T.; Sagun, L.; and Zecchina, R. 2017. Entropy-SGD: Biasing Gradient Descent Into Wide Valleys. In *5th International Conference on Learning Representations, 2017*.

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

Correia, A. H.; Gala, G.; Quaeghebeur, E.; De Campos, C.; and Peharz, R. 2023. Continuous mixtures of tractable probabilistic models. In *AAAI Conference on Artificial Intelligence*, volume 37, 7244–7252.

Dang, M.; Liu, A.; and den Broeck, G. V. 2022. Sparse Probabilistic Circuits via Pruning and Growing. In Koyejo, S.; Mohamed, S.; Agarwal, A.; Belgrave, D.; Cho, K.; and Oh, A., eds., *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems, 2022*.

Darwiche, A. 2003. A Differential Approach to Inference in Bayesian Networks. *Journal of the ACM*, (3): 280–305.

Desana, M.; and Schnörr, C. 2017. Learning Arbitrary Sum-Product Network Leaves with Expectation-Maximization. arXiv:1604.07243.

Dos Martires, P. Z. 2024. Probabilistic neural circuits. In *AAAI Conference on Artificial Intelligence*, volume 38, 17280–17289.

Foret, P.; Kleiner, A.; Mobahi, H.; and Neyshabur, B. 2021. Sharpness-aware Minimization for Efficiently Improving Generalization. In *9th International Conference on Learning Representations, 2021*.

Gal, Y.; and Ghahramani, Z. 2016. Dropout as a Bayesian Approximation: Representing Model Uncertainty in Deep Learning. In Balcan, M.; and Weinberger, K. Q., eds., *33rd International Conference on Machine Learning, 2016*, JMLR Workshop and Conference Proceedings, 1050–1059.

Gala, G.; de Campos, C. P.; Peharz, R.; Vergari, A.; and Quaeghebeur, E. 2024. Probabilistic Integral Circuits. In Dasgupta, S.; Mandt, S.; and Li, Y., eds., *International Conference on Artificial Intelligence and Statistics, 2024*, Proceedings of Machine Learning Research, 2143–2151.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative Adversarial Nets. In Ghahramani, Z.; Welling, M.; Cortes, C.; Lawrence, N. D.; and Weinberger, K. Q., eds., *Advances in Neural Information Processing Systems 27: Annual Conference on Neural Information Processing Systems, 2014*, 2672–2680.

Hochreiter, S.; and Schmidhuber, J. 1997. Flat minima. *Neural computation*, (1): 1–42.

Hutchinson, M. F. 1990. A stochastic estimator of the trace of the influence matrix for laplacian smoothing splines. *Communications in Statistics - Simulation and Computation*.

Karanam, A.; Mathur, S.; Sidheekh, S.; and Natarajan, S. 2025. A Unified Framework for Human-Allied Learning of Probabilistic Circuits. In *AAAI Conference on Artificial Intelligence, 2025*, volume 39, 17779–17787.

Kaur, S.; Cohen, J.; and Lipton, Z. C. 2023. On the Maximum Hessian Eigenvalue and Generalization. In *Proceedings of Machine Learning Research*.

Keskar, N. S.; Mudigere, D.; Nocedal, J.; Smelyanskiy, M.; and Tang, P. T. P. 2017. On Large-Batch Training for Deep Learning: Generalization Gap and Sharp Minima. In *5th International Conference on Learning Representations, 2017*.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In Bengio, Y.; and LeCun, Y., eds., *2nd International Conference on Learning Representations, 2014*.

Kisa, D.; Broeck, G. V. D.; Choi, A.; and Darwiche, A. 2014. Probabilistic sentential decision diagrams. In *International Conference on Knowledge Representation and Reasoning*.

Kwon, J.; Kim, J.; Park, H.; and Choi, I. K. 2021. ASAM: Adaptive Sharpness-Aware Minimization for Scale-Invariant Learning of Deep Neural Networks. In Meila, M.; and Zhang, T., eds., *38th International Conference on Machine Learning, 2021*, Proceedings of Machine Learning Research, 5905–5914.

Li, H.; Xu, Z.; Taylor, G.; Studer, C.; and Goldstein, T. 2018. Visualizing the Loss Landscape of Neural Nets. In Bengio, S.; Wallach, H. M.; Larochelle, H.; Grauman, K.; Cesa-Bianchi, N.; and Garnett, R., eds., *Advances in Neural Information Processing Systems 31: Annual Conference on Neural Information Processing Systems 2018*, 6391–6401.

Liu, A.; Ahmed, K.; and den Broeck, G. V. 2024. Scaling Tractable Probabilistic Circuits: A Systems Perspective. In *Forty-first International Conference on Machine Learning, 2024*.

Liu, A.; and den Broeck, G. V. 2021. Tractable Regularization of Probabilistic Circuits. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, 2021*, 3558–3570.

Liu, A.; Niepert, M.; and den Broeck, G. V. 2024. Image Inpainting via Tractable Steering of Diffusion Models. In *The Twelfth International Conference on Learning Representations, 2024*.

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, 2023*.

Loconte, L.; Mari, A.; Gala, G.; Peharz, R.; de Campos, C.; Quaeghebeur, E.; Vessio, G.; and Vergari, A. 2025. What is the Relationship between Tensor Factorizations and Circuits (and How Can We Exploit it)? *Transactions on Machine Learning Research*.

Loconte, L.; Mengel, S.; and Vergari, A. 2025. Sum of squares circuits. In *AAAI Conference on Artificial Intelligence*, 18, 19077–19085.

Loconte, L.; Sladek, A. M.; Mengel, S.; Trapp, M.; Solin, A.; Gillis, N.; and Vergari, A. 2024. Subtractive Mixture Models via Squaring: Representation and Learning. In *The Twelfth International Conference on Learning Representations, 2024*.

Maene, J.; and De Raedt, L. 2025. The Gradient of Algebraic Model Counting. In *AAAI Conference on Artificial Intelligence*, volume 39, 19367–19377.

Martens, J.; Sutskever, I.; and Swersky, K. 2012. Estimating the Hessian by Back-propagating Curvature. In *29th International Conference on Machine Learning, 2012*.

Papamakarios, G.; Nalisnick, E. T.; Rezende, D. J.; Mohamed, S.; and Lakshminarayanan, B. 2021. Normalizing Flows for Probabilistic Modeling and Inference. *Journal of Machine Learning Research*, 57:1–57:64.

Peharz, R.; Lang, S.; Vergari, A.; Stelzner, K.; Molina, A.; Trapp, M.; den Broeck, G. V.; Kersting, K.; and Ghahramani, Z. 2020. Einsum Networks: Fast and Scalable Learning of Tractable Probabilistic Circuits. In *37th International Conference on Machine Learning, 2020*, Proceedings of Machine Learning Research, 7563–7574.

Peharz, R.; Vergari, A.; Stelzner, K.; Molina, A.; Trapp, M.; Shao, X.; Kersting, K.; and Ghahramani, Z. 2019. Random Sum-Product Networks: A Simple and Effective Approach to Probabilistic Deep Learning. In Globerson, A.; and Silva, R., eds., *Thirty-Fifth Conference on Uncertainty in Artificial Intelligence, 2019*, Proceedings of Machine Learning Research, 334–344.

Poon, H.; and Domingos, P. M. 2011. Sum-Product Networks: A New Deep Architecture. In Cozman, F. G.; and Pfeffer, A., eds., *Twenty-Seventh Conference on Uncertainty in Artificial Intelligence, 2011*, 337–346.

Rahman, T.; Kothalkar, P.; and Gogate, V. 2014. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *Machine Learning and Knowledge Discovery in Databases: European Conference, 2014*, 630–645. Springer.

Sankar, A. R.; Khasbage, Y.; Vigneswaran, R.; and Balasubramanian, V. N. 2021. A Deeper Look at the Hessian Eigenspectrum of Deep Neural Networks and its Applications to Regularization. In *Thirty-Fifth AAAI Conference on Artificial Intelligence, 2021, Thirty-Third Conference on Innovative Applications of Artificial Intelligence, The Eleventh Symposium on Educational Advances in Artificial Intelligence*, 9481–9488.

Shao, X.; Molina, A.; Vergari, A.; Stelzner, K.; Peharz, R.; Liebig, T.; and Kersting, K. 2022. Conditional sum-product networks: Modular probabilistic circuits via gate functions. *International Journal of Approximate Reasoning*, 140: 298–313.

Sharir, O.; and Shashua, A. 2018. Sum-product-quotient networks. In *International Conference on Artificial Intelligence and Statistics*, 529–537. PMLR.

Shih, A.; Sadigh, D.; and Ermon, S. 2021. HyperSPNs: Compact and Expressive Probabilistic Circuits. In Ranzato, M.; Beygelzimer, A.; Dauphin, Y. N.; Liang, P.; and Vaughan, J. W., eds., *Advances in Neural Information Processing Systems 34: Annual Conference on Neural Information Processing Systems, 2021*, 8571–8582.

Sidheekh, S.; Kersting, K.; and Natarajan, S. 2023. Probabilistic Flow Circuits: Towards Unified Deep Models for Tractable Probabilistic Inference. In Evans, R. J.; and Shpitser, I., eds., *Uncertainty in Artificial Intelligence, 2023*, Proceedings of Machine Learning Research, 1964–1973.

Sidheekh, S.; and Natarajan, S. 2024. Building Expressive and Tractable Probabilistic Generative Models: A Review. In *Thirty-Third International Joint Conference on Artificial Intelligence, 2024*, 8234–8243.

Sidheekh, S.; Tenali, P.; Mathur, S.; Blasch, E.; Kersting, K.; and Natarajan, S. 2025. Credibility-Aware Multimodal Fusion Using Probabilistic Circuits. In *The 28th International Conference on Artificial Intelligence and Statistics*.

Suresh, H.; Sidheekh, S.; Natarajan, S.; Krishnan, N. C.; et al. 2025. Tractable Sharpness-Aware Learning of Probabilistic Circuits. *arXiv preprint arXiv:2508.05537*.

Ventola, F.; Braun, S.; Yu, Z.; Mundt, M.; and Kersting, K. 2023. Probabilistic circuits that know what they don't know. In Evans, R. J.; and Shpitser, I., eds., *Uncertainty in Artificial Intelligence, 2023*, Proceedings of Machine Learning Research, 2157–2167.

Vergari, A.; Choi, Y.; Liu, A.; Teso, S.; and Van den Broeck, G. 2021. A compositional atlas of tractable circuit operations for probabilistic inference. *Advances in Neural Information Processing Systems*, 34: 13189–13201.

Wang, B.; and Van den Broeck, G. 2025. On the relationship between monotone and squared probabilistic circuits. In *AAAI Conference on Artificial Intelligence*, 20, 21026–21034.

Zhang, H.; Dang, M.; Peng, N.; and den Broeck, G. V. 2023. Tractable Control for Autoregressive Language Generation. In *International Conference on Machine Learning,2023*, Proceedings of Machine Learning Research, 40932–40945.

Zhang, H.; Dang, M.; Wang, B.; Ermon, S.; Peng, N.; and Van den Broeck, G. 2025. Scaling Probabilistic Circuits via Monarch Matrices. In *42nd International Conference on Machine Learning*.