A Quantum Information Theoretic Approach to Tractable Probabilistic Models

Anonymous Authors[1]

Abstract

By recursively nesting sums and products, probabilistic circuits (PCs) have emerged in recent years as an attractive class of generative models as they enjoy, for instance, polytime marginalization of random variables. However, as these PCs form monotone functions, specifically the parameters are constrained to positive real values, their expressive power is limited. Drawing inspiration quantum information theory we introduce *positive unitary circuits* (PUnCs), which generalize circuit evaluations over positive real-valued probabilities to circuit evaluation over positive semidefinite matrices. A key feature of PUnCs is that they form normalized probability distributions by construction, which avoids the computation of a spurious normalization constant. Motivated by noisy quantum systems we generalize PUnCs to NoisePUnCs, which can be interpreted as deep mixtures of PUnCs. Lastly, we provide an efficient parametrization of PUnCs that enables us to perform parameter learning on hardware accelerators using standard and unconstrained gradient-based optimization.

1. Introduction

Probabilistic circuits (PCs) (Darwiche, 2003; Poon & Domingos, 2011) belong to an unusual class of probabilistic models: they are highly expressive but at the same time also tractable. For instance, so-called decomposable probabilistic circuits (Darwiche, 2001a) allow for the computation of marginals in time polynomial in the size of the circuit. Zhang et al. (2020) noted that it is exactly this restriction to positive values that limits the expressive efficiency (or succinctness) of PCs (Martens & Medabalimi, 2014; de Colnet & Mengel, 2021). In particular, the positivity constraint on the set of elements that PCs operate on prevents them from modelling negative correlations between variables. Circuits that are incapable of modelling negative correlations, i.e. circuits that can only combine probabilities in an additive fashion, are also called monotone circuits (Shpilka & Yehudayoff, 2010). This restricted expressiveness can be combatted by the use of so-called *non-monotone* circuits, where subtractions are allowed as a third operation (besides sums and products). Interestingly, Valiant (1979) showed that a mere single subtraction can render non-monotone circuits exponentially more expressive than monotone circuits a result that has recently been refined for decomposable circuits (Loconte et al., 2024b).

As shown in (Harviainen et al., 2023; Agarwal & Bläser, 2024), non-monotone circuits do, however, introduce an important complication: if non-monotone circuits are not designed carefully, verifying whether a circuit encodes a valid probability distribution or not is an NP-hard problem. This does also render learning the parameters of a circuit practically infeasible.

Using the concept of *positive operator valued measures* from quantum information theory, which encode random event as positive semidefinite matrices we are able to construct non-monotone circuits that nonetheless encode proper (normalized) probability distributions by construction. Our work falls into a line of recent works presented in the circuit literature (Sladek et al., 2023; Loconte et al., 2024c; Wang & Van den Broeck, 2024; Loconte et al., 2024b). However, our work is the first that establishes this deep connection between concepts in quantum information theory and tractable probabilistic models. We make three concrete contributions 1. We generalize probabilistic circuits over scalars to operator-valued circuits and introduce PUnCs (Section 3). 2. We use the concept of noise in quantum information theory to construct deep operator mixture circuits, dubbed NoisePUnCs (Section 4). 3. We provide an efficient parametrization such that the constructed probability distributions using PUnCs are normalized by construction (Section 5.2).

2. A Primer on Quantum Information Theory

A widely used and elegant framework to describe measurements of quantum systems is the so-called *positive operator-valued measure* (POVM) formalism. While POVMs have

[1]Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

physical interpretations in terms of quantum information and quantum statistics, we will only be interested in their mathematical prperties as we use them to show that circuits (defined in Section 3) form valid probability distributions. We refer the reader to (Nielsen & Chuang, 2001) for an in-depth exposition on the topic, as well as quantum computing and quantum information theory in general.

Definition 2.1 (Positive Semidefinite)**.** An $B \times B$ Hermitian matrix H is called positive semidefinite (PSD) if and only if $\forall \mathbf{x} \in \mathbb{C}^B : \mathbf{x}^* H \mathbf{x} \geq 0$, where $\mathbf{x}^*$ denotes the conjugate transpose and $\mathbb{C}^B$ the B-dimensional space of complex numbers.

Definition 2.2 (POVM (Nielsen & Chuang, 2001, Page 90))**.** A positive operator-valued measure is a set of PSD matrices $\{E(i)\}_{i=0}^{I-1}$ (I being the number of possible measurement outcomes) that sum to the identity:

$$\textstyle\sum_{i=0}^{I-1} E(i) = \mathbb{1}, \tag{1}$$

Before defining the probability of a specific i occurring, we need the notion of a density matrix (von Neumann, 1927):

Definition 2.3 (Density Matrix (Nielsen & Chuang, 2001, Page 102))**.** A density matrix ρ is a PSD matrix of trace one, i.e. $\mathrm{Tr}[\rho] = 1$.

Definition 2.4 (Event Probability (Nielsen & Chuang, 2001, Page 102))**.** Let ρ be a density matrix and let i denote an event with $E(i)$ being the corresponding element from the POVM. The probability of the event i happening, i.e. measuring the outcome i, is given by

$$p(i) = \mathrm{Tr}[\rho E(i)] \tag{2}$$

Proposition 2.5. *The expression in Equation 2 defines a valid probability distribution.*

Proof. While this is a well-known result we were not able to identify a concise proof in the literature. We therefore provide one in Appendix A.1. □

out Loewner

(Baksalary et al., 1989, Corollary 4.2)

3. Positive Unitary Circuits

A popular subclass of probabilistic circuits are so-called structured decomposable probabilistic circuits (Darwiche, 2011) that are also smooth (Darwiche, 2001b). The advantage of this circuit subclass is that they can be implemented in a rather straightforward fashion on modern AI accelerators, as demonstrated by Peharz et al. (2019; 2020). Zuidberg Dos Martires (2024) introduced an abstraction for these smooth structured decomposable circuit dubbed *partition circuits*. We give such a circuit in Figure 1.

E paper

e paper

Definition 3.1 (Partition Circuit (?))**.** A partition circuit over a set of variables is a parametrized computation graph taking

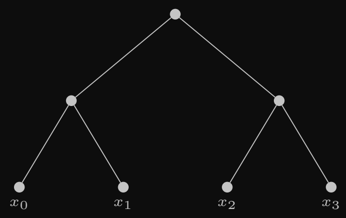

Figure 1: Partition circuit over four binary variables x_i with $i \in \{0, 1, 2, 3\}$, which are given as inputs to the circuit. The internal nodes of the partition circuit correspond the computation units.

the form of a binary tree. The partition circuit consists of two kinds of computation units: *leaf* and *internal* units, as well as a single *root*. Units at the same distance from the root form a layer. Furthermore, let p_k denote the root unit or an internal unit. The unit p_k then receives its inputs from two units in the previous layer, which we denote by p_{k_l} and p_{k_r}. Each computation unit is input to exactly one other unit, except the root unit, which is the input to no other unit.

3.1. Operator Circuits

Using the concept of partition circuits we now generalize the probabilistic circuits to (unnormalized) positive operator circuits. Positive operator circuits can be thought of as generalizing circuit evaluations with probabilities to circuit evaluations with PSD matrices. Note, in the definition below we use O_k instead of p_k to make this generalization explicit.

Definition 3.2 (Operator Circuit)**.** Let $\mathbf{x} = \{x_0, \dots, x_{N-1}\}$ be a set of M categorical variables with domains of size S. We define an operator circuit as a partition circuit whose computation units take the following functional form:

$$O_k(\mathbf{x}_k) = \begin{cases} A_k \times e_{x_k} \otimes e^*_{x_k} \times A^*_k, & \text{if } k \text{ is leaf} \\ \mathscr{B}_k\Big(O_{k_l}(\mathbf{x}_{k_l}), O_{k_l}(\mathbf{x}_{k_l})\Big) & \text{else} \end{cases} \tag{3}$$

Here we use the symbol $\otimes$ to denote the Kronecker product. Furthermore, $\{e_{x_k}\}_{n=0}^{S-1}$ is a set of S-dimensinoal orthonormal basis vectors associating each of the S possible values for x_k to a specific basis vector. The functions $\mathscr{B}_k(\cdot, \cdot)$ denote bilinear forms.

Definition 3.3. We call an operator circuit *positive* if we have that the bilinear forms $\mathscr{B}_k$ are such that

$$\mathscr{B}_k\Big(O_{k_l}(\mathbf{x}_{k_l}), O_{k_l}(\mathbf{x}_{k_l})\Big) \quad \text{is a PSD matrix} \tag{4}$$

if $O_{k_l}(\mathbf{x}_{k_l})$ and $O_{k_l}(\mathbf{x}_{k_l})$ are PSD.

Proposition 3.4. *Positive operator circuits are PSD.*

Proof. See Appendix A.2. □

3.2. Constructing a Probability Distribution

In Section 2 we saw that we can construct a probability distribution using a density matrix ρ and a positive operator-valued measure, with the latter being a set of PSD matrices (cf. Definition 2.2) that sum to the unit matrix. Using a positive operator circuit $O(\mathbf{x})$ we indeed have a set of PSD matrices. Namely, on for each instantiation of the $\mathbf{x}$ variables. We now introduce *positive unit preserving operator circuits* for which also the summation to the unit matrix holds.

Definition 3.5. We call a positive operator circuit *unit preserving* if we have that the bilinear forms $\mathscr{B}_k$ are such that

$$\mathscr{B}_k(\mathbb{1}_{k_l}, \mathbb{1}_{k_r}) = \mathbb{1}_k, \tag{5}$$

where $\mathbb{1}_k$, $\mathbb{1}_{k_l}$, and $\mathbb{1}_{k_r}$ denote unit matrices of appropriate size, and if $A_k A_k^* = \mathbb{1}_k$ in the leaves. That is, the A_k are semi-unitary matrices.

We will also refer to positive unit preserving operator circuits as positive unitary circuits, or PUnCs.

Proposition 3.6. *Let $\mathbf{X}$ denote a set of random variables with sample space $\Omega(\mathbf{X})$. Then the set $\{\widehat{O}(\mathbf{x})\}_{\mathbf{x}\in\Omega(\mathbf{X})}$ of positive unitary circuis forms a POVM.*

Proof. See Appendix A.3 □

Theorem 3.7. *Let ρ be a density matrix and $\widehat{O}(\mathbf{x})$ a positive unitary circuit. The function*

$$p_{\mathbf{X}}(\mathbf{x}) = \mathrm{Tr}\left[\widehat{O}(\mathbf{x})\rho\right] \tag{6}$$

is a proper probability distribution over the random variables $\mathbf{X}$ with sample space $\Omega(\mathbf{X})$.

Proof. This follows from Proposition 2.5 and Proposition 3.6 □

4. Quantum Noise and Circuits

4.1. Sub-Complete Probability Distributions

An important concept in quantum information theory that generalizes the standard POVM (cf. Definition 2.2) is the so-called noisy POVM, which enables for instance modelling imperfect measurement devices. In the context of machine learning this might be an improperly labeled data point of an error in the detector, e.g. a pixel flip in the camera that took a picture.

Definition 4.1. A noisy positive operator-valued measure is a set of PSD matrices $\{E(i)\}_{i=0}^{I-1}$ (I being the number of possible measurement outcomes) such that:

$$\textstyle\sum_{i=0}^{V-1} E(i) = M < \mathbb{1}, \tag{7}$$

where the inequality sign is interpreted using the Loewner ordering of PSD matrices, i.e. $M < \mathbb{1} \Leftrightarrow \mathbb{1} - M$ is PSD.

Definition 4.2 (Sub-complete Probability Distribution)**.** Let $\mathbf{X}$ be a set of random variables with sample space $\Omega(\mathbf{X})$. We call a probability distribution *sub-complete* if $\forall \mathbf{x} \in \Omega(\mathbf{X}) : p(\mathbf{x}) \geq 0$ and if $\sum \mathbf{x} \in \Omega(\mathbf{X}) p(\mathbf{x}) \leq 1$. We call a probability distribution *striclty sub-complete* if the latter inequality holds strictly and *complete* if equality holds.

Proposition 4.3. *Noisy POVMs induce a sub-complete probability distributions.*

Proof. See Appendix 4.3 □

To construct sub-complete probability distributions using operator circuits we now introduce *noisy positive unit preserving operator circuits* or NoisePUnCs.

Definition 4.4. A NoisePUnC $Q(\mathbf{x})$ is of the form

$$Q(\mathbf{x}) = q(\mathbf{x})\widehat{O}(\mathbf{x}), \tag{8}$$

where $\widehat{O}(\mathbf{x})$ is a positive unit preserving operator circuit and $q(\mathbf{x})$ is real-valued and belongs to the $[0, 1]$ interval for every $\mathbf{x}$.

Proposition 4.5. *NoisePUnCs induce sub-complete probability distributions.*

Proof. See Appendix A.5 □

4.2. Noise Variables

All of this is a bit unnerving. How is it possible that we have probabilistic events $\mathbf{x} \in \Omega(\mathbf{X})$ that violate the second of Kolmogorov's probability axioms (by having sub-complete distributions)? The resolution to this problem lies in interpreting sub-complete distributions as joint distributions not only over the set of variables $\mathbf{X}$ but an extra set of variables $\mathbf{Y}$ representing extra noise (Wiseman & Milburn, 2009)

$$\pi_{\mathbf{X}}(\mathbf{x}) = p_{\mathbf{XY}}(\mathbf{x}, \mathbf{y}), \tag{9}$$

for which we indeed have

$$\textstyle\sum_{\mathbf{x}\in\Omega(\mathbf{X})} \sum_{\mathbf{y}\in\Omega(\mathbf{Y})} p_{\mathbf{XY}}(\mathbf{x}, \mathbf{y}) = 1. \tag{10}$$

The problem with the random variables $\mathbf{Y}$ is that they are not accessible to the observer. In the sense that they are observable in principle but in practice not, e.g. a defect in a sensor.[1] However, the probability we are actually interested in is the one for $\mathbf{X}$ given $\mathbf{Y}$

$$\begin{aligned} p_{\mathbf{X}|\mathbf{Y}}(\mathbf{x} \mid \mathbf{y}) &= {p_{\mathbf{XY}}(\mathbf{x},\mathbf{y})}/{\textstyle\sum_{\mathbf{x}\in\Omega(\mathbf{X})} p_{\mathbf{XY}}(\mathbf{x},\mathbf{y})} \\ &= {\pi_{\mathbf{X}}(\mathbf{x})}/{\textstyle\sum_{\mathbf{x}\in\Omega(\mathbf{X})} \pi_{\mathbf{X}}(\mathbf{x})} \end{aligned} \tag{11}$$

The question now became whether we can choose $q(\mathbf{x})$ such that the summation in the denominator is tractable, i.e. can be performed in time polynomial in the number of random variables $\mathbf{X}$.

[1] Note these practically unobservable variables should not be confused with the concept of (local) hidden variables, which lead to violations of Bell's inequality.

babislitc cir-
nstead of soft
ng seems to
etter

4.3. Soft Counting Circuits

In the probabilistic circuit literature in has been shown that the product of two circuits is representable as a single circuit within polytime and polyspace if the two circuits are compatible (Khosravi et al., 2019; Vergari et al., 2021). Importantly, this single circuit then allows for summing out the variables $\mathbf{x}$. We will now show that this also holds when the $q(\mathbf{x})$ is a circuit and $\widehat{O}(\mathbf{x})$ is a PUnC, and both share the same partition tree. To this end we also introduce a new circuit type that satisfies the $0 \leq q(\mathbf{x}) \leq 1$ constraint imposed in Definition 4.4.

Definition 4.6 (Soft Counting Circuit)**.** Let $\mathbf{x}=\{x_0, \dots, x_{N-1}\}$ be a set of N categorical variables with domains of size S. We define an operator circuit as a partition circuit whose computation units take the following functional form:

$$q_k(\mathbf{x}_k) = \begin{cases} W_k \times \text{one\_hot}(x_k), & \text{if } k \text{ leaf} \\ W_k \times \left(q_{k_l}(\mathbf{x}_{k_l}) \odot q_{k_r}(\mathbf{x}_{k_r}) \right) & \text{else.} \end{cases}$$

Here we use the symbols $\times$ and $\odot$ to denote the matrix product and Hadamard product, respectively. In the leaves we have $W_k \in \mathbb{R}_{\geq 0}^{C \times S}$ with C being a positive integer determining the capacity of the circuit. We also require the entries of W_k in the leaves to be upper bounded by 1. For the W_k in the internal units and the root unit we require their entries to be positive reals and that they are row-normalized. That is, $\forall k, i : \sum_j W_{kij} = 1$, where the i and j indices index the matrix. For internal units we have $W_k \in \mathbb{R}_{\geq 0}^{C \times C}$ and for the root we have $W_{root} \in \mathbb{R}_{\geq 0}^{1 \times C}$.

For the sake of simplicity we have chosen here a canonical polyadic form (Carroll & Chang, 1970) for the internal units by using a Hadamard product to decompose variable sets. We refer the reader to (Loconte et al., 2024a) for discussions on alternative decomposition strategies.

Proposition 4.7. *A soft counting circuit $q(\mathbf{x})$ satisfies $0 \leq q(\mathbf{x}) \leq 1$.*

Proof. See Appendix A.6 □

We note that soft counting circuits share many similarities with probabilistic circuits, in that they are both monotone by having weights matrices W_k that have positive entries only. Furthermore, both circuit types have row-normalized weight matrices in the internal units and the root unit. The only difference, really, is that in the leaves, probabilistic circuits have row-normalized weight matrices as well, while the entries of weight matrices of soft counting circuits are bounded between zero and one. This has as a consequence that for a probabilistic circuit $p(\mathbf{x})$ we have $\sum_{\mathbf{x} \in \Omega(\mathbf{X})} p(\mathbf{x}) = 1$, while we have for soft counting circuits $0 \leq \sum_{\mathbf{x} \in \Omega(\mathbf{X})} p(\mathbf{x}) \leq S^N$. With N being the number of variables and S being the domain size of the variables.

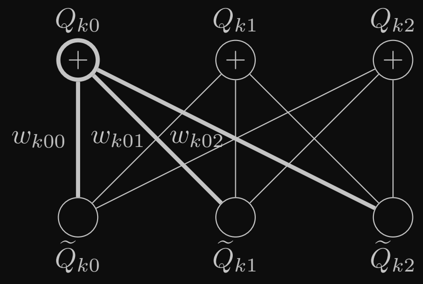

Figure 2: Graphical representation of operator mixing as described in Equation 14. The operators Q_{k0}, Q_{k1}, and Q_{k2} are all PSD matrices and constructed using the operators $\widetilde{Q}_{k0}$, $\widetilde{Q}_{k1}$, and $\widetilde{Q}_{k2}$ using weighted sums. For Q_{k0} we also indicate the mixing weights w_{k00}, w_{k01}, and w_{k02}, which are positive real-valued scalars and satisfy $w_{k00} + w_{k01} + w_{k02} = 1$.

4.4. NoisePUnCs as Deep Operator Mixtures

An interesting consequence of modelling noise in the system via a soft counting circuit $q(\mathbf{x})$ is that it effectively results in a model that can be interpreted as a recursive mixture of positive operators. To see this, let us first write the matrix vector product in the computation units of $q(\mathbf{x})$ using explicit indices:

$$q_k = W_k \times (q_{k_l} \odot q_{k_r}) \Leftrightarrow q_{ki} = \textstyle\sum_j w_{kij} q_{k_l j} q_{k_r j}. \quad (12)$$

Here q_{ki} denotes the i-th entry of the vector q_k and w_{kij} denotes the elements of the matrix W_k. We also dropped the explicit dependency on $\mathbf{x}_k$.

We now multiply each $q_{ki} \in [0, 1]$ with the corresponding operator $\widehat{O}_k$

$$\begin{aligned} q_{ki}\widehat{O}_k &= \left(\textstyle\sum_j w_{kij} q_{k_l j} q_{k_r j} \right) \mathscr{B}_k(\widehat{O}_{k_l}, \widehat{O}_{k_r}) \\ &= \textstyle\sum_j w_{kij} \underbrace{\mathscr{B}_k(q_{k_l j}\widehat{O}_{k_l}, q_{k_r j}\widehat{O}_{k_r})}_{=\widetilde{Q}_{kj}} \end{aligned} \quad (13)$$

This means that we have for each computation unit k not one but multiple operators, indexed by i, and each of the i operators is a mixture of operators:

$$Q_{ki}(\mathbf{x}_k) = \textstyle\sum_j w_{kij} \widetilde{Q}_{kj}(\mathbf{x}_k). \quad (14)$$

We illustrate the situation in Figure 2. Such mixtures of operators can be regarded as generalizing standard mixture models of probability distributions, which can be recovered by considering the special case of operators of dimension one, in other words positive real-valued scalars.

Recursively nesting such weighted sums of operators allows for representing a polynomial with exponentially many terms (Martens & Medabalimi, 2014). However, in practice we only need to materialize these mixtures when computing $\sum_{\mathbf{x}\in\Omega(\mathbf{X})\pi_{\mathbf{X}}(\mathbf{x})}$ in order to normalize the probabilities. If we evaluate a NoisePUnC it suffices to evaluate the soft counting circuit $q(\mathbf{x})$ and the PUnC $\widehat{O}(\mathbf{x})$ separately, and use the individual circuit evaluations to compute the unnormalized but bounded probability.

Loconte et al. (2024a) proposed a similar combination of monotone and non-monotone circuits. This was motivated by their observation that the expressive power of monotone and squared circuits (a special case of positive operator circuits) are incomparable (de Colnet & Mengel, 2021). That is, each of these circuit classes can express circuits that would lead to an exponential blow-up in the respective other circuit class. Interestingly, we arrived at a similar construction by starting from noise in quantum circuits.

5. Parametrizing PUnCs

In order to parametrize positive unitary circuits, we need to find a functional form of the bilinear forms such that Equation 4 and Equation 5 hold, and such that we have $A_k A_k^* = \mathbb{1}$. For the latter we simply pick A_k to be the (possibly truncated) orthonormal discrete Fourier transform matrix. This means that the A_k are identical for all the leaves and that the leaves are parameter free but guarantees that we indeed have $A_k A_k^* ? \mathbb{1}$.

For the bilinear forms we take inspiration from the probabilistic circuit literature and use a canonical polyadic ansatz (Carroll & Chang, 1970).

$$\mathscr{B}_k(O_{k_l}, O_{k_l}) = L_k O_{k_l} L_k^* \odot R_k O_{k_r} R_k^*, \qquad (15)$$

where L_k and R_k are complex valued matrices satisfying $L_k L_k^* \odot R_k R_k^* = \mathbb{1}_{B\times B}$.

Assuming O_{k_l} and O_{k_r} are PSD we also have that $L_k O_{k_l} L_k^*$ and $R_k O_{k_r} R_k^*$ are PSD. Furthermore, using Schur's product theorem (Schur, 1911, p. 14, Theorem VII) that states that the Hadamard product of two PSD matrices is again PSD lets us conclude that parametrizing operator circuits with bilinear forms from Equation 15 results in the circuit being positive. For a discussion on the constraint $L_k L_k^* \odot R_k R_k^* = \mathbb{1}_{B\times B}$ we point to Section 5.2.

Proposition 5.1. *Let N be the number of variables $\mathbf{x}_k$, having all the domain size S, and let the L_k and R_k matrices be $B \times B$ square matrices, with $B \leq S$. We can perform marginal inference in PUnCs in $\mathcal{O}(NS^3)$ when using the bilinear form from Equation 15.*

Proof. See Append A.7 □

5.1. The Vector Representation

The matrix-matrix multiplication present in Equation 15 lead to the cubic computation cost per computation unit $\mathcal{O}(S^3)$ (cf. Section A.7). By exploiting the particular structure of the bilinear form in Equation 15 and by rearranging the operations performed in the operator circuit, we can avoid these matrix-matrix multiplications entirely and obtain matrix-vector multiplications instead. We call this alternative representation the *vector* representation of an operator circuit.

Definition 5.2 (Vector Representation)**.** Let $\mathbf{x}=\{x_0, \ldots, x_{N-1}\}$ be a set of N categorical variables with domains of size S. We define the vector representation of an operator circuit as a partition circuit whose computation units take the following functional form:

$$v_k(\mathbf{x}_k) = \begin{cases} A_k \times e_{x_k}, & \text{if } k \text{ leaf} \\ L_k \times v_{k_l}(\mathbf{x}_k) \odot R_k \times v_{k_r}(\mathbf{x}_k), & \text{else} \end{cases} \qquad (16)$$

Proposition 5.3. *The expression $\|\gamma \times v_{root}(\mathbf{x})\|^2$, with $\rho = \gamma \times \gamma^*$ computes the same probability as $\mathrm{Tr}[O_{root}(\mathbf{x})\rho]$.*

Proof. See Section A.8 □

Proposition 5.4. *PUnCs in their vector representation compute joint probabilities in time $\mathcal{O}(NS^2)$*

Proof. See Section A.9 □

Note that the vector representation can only be used for computing joint probabilities as marginalizing out variables forces us to go to the matrix representation of operator circuits. Note also that the vector representation of PUnCs can be seen as a PSD circuits (Sladek et al., 2023) or equivalently sum of squares circuits (Loconte et al., 2024b), which we obtained by imposing a specific functional structure on the bilinear forms.

Having an efficient evaluation for PUnCs in their vector representation also means that we can evaluate NoisePUnCs more efficiently as we can compute $q_{root}(\mathbf{x})$ and $\pi_{\mathbf{X}}(\mathbf{x})$ separately, with the former having a computational complexity of $\mathcal{O}(NCS + NC^2)$. Computing the normalization constant for a NoisePUnC is more involved, and we discuss it along with its computational complexity in Appendix B. However, we note that it is still feasible in polytime.

5.2. An Efficient Parametrization

Having specified the functional form of the positive unitary circuits using polyadic bilinear forms we would also like to find an efficient parametrization of the γ, L_k, and R_k matrices that is amenable to unconstrained gradient descent optimization. For γ this is rather straightforward as we can simply use: $\gamma = \frac{\tilde{\gamma}}{\sqrt{\mathrm{Tr}[\tilde{\gamma}\tilde{\gamma}^*]}}$, where $\tilde{\gamma} \in \mathbb{C}^{B\times B}$ is an arbitrary complex values matrix. Note that we have almost by definition $\mathrm{Tr}[\rho] = 1$.

For the L_k and R_k matrices we make some further specifying choices. Consider the real-valued diagonal matrix $D_k \in \mathbb{R}^{\mathbb{B}\times\mathbb{B}}$ and the two unitary matrices $U_k \in \mathbb{C}^{\mathbb{B}\times\mathbb{B}}$ and $V_k \in \mathbb{C}^{\mathbb{B}\times\mathbb{B}}$, which we use to construct the L_k and R_k matrices:

$$L_k = D_k^{1/2} U_k, \quad \text{and} \quad R_k = D_k^{-1/2} V_k. \tag{17}$$

We can verify that this parametrization satisfies the unit preserving condition of for the bilinear form (cf. Definition 3.5).

$$\begin{aligned} B_k(\mathbb{1}_{B\times B}, \mathbb{1}_{B\times B}) &= L_k \mathbb{1}_{B\times B} L_k^* \odot R_k \mathbb{1}_{B\times B} R_k^* \\ &= D_k^{1/2} U_k U_k^* D_k^{1/2} \odot D_k^{-1/2} V_k V_k^* D_k^{-1/2} \\ &= \mathbb{1}_{B\times B} \end{aligned} \tag{18}$$

While we now have a prescription to parametrize PUnCs, one problem remains: unitary matrices are notoriously difficult to parametrize efficiently (Kiani et al., 2022; Jing et al., 2017; Lezcano-Casado & Martınez-Rubio, 2019; Mhammedi et al., 2017; Wisdom et al., 2016).

We will take a practical approach here and parametrize the unitary matrices using a product of a diagonal and circulant matrix:

$$U_k = \Lambda_{U_k} F_B \Sigma_{U_k} F_B^* \quad \text{and} \quad V_k = \Lambda_{V_k} F_B \Sigma_{V_k} F_B^*. \tag{19}$$

Here Λ_{U_k}, Λ_{V_k}, Σ_{U_k}, and Σ_{V_k} are diagonal matrices whose entries have no magnitude, i.e. they are of the form $e^{i\phi}$ with $\phi \in \mathbb{R}$. Furthermore, F_B denotes the orthonormalized B-point discrete Fourier transform matrix, and we exploit the fact any circulant $B \times B$ matrix can be written as a diagonal matrix sandwidched between F_B and F_B^*. Recalling that F_B is unitary it is again straightforward to show that this parametrization yields unitary matrices. We also refer the interested reader to Araujo et al. (2020) for a in depth discussion of diagonal-circulant parametrization of neural networks.

While using this diagonal-circulant parametrization does not cover the entire space of unitary matrices it has the advantage of being parametrizable with unconstrained parameters. Furthermore, using the fact that mulitplying a discrete Fourier transform matrix with a vector can be done on $\mathcal{O}(B \log B)$ brings down the computational cost of computing the bilinear form from $\mathcal{O}(B^2)$.

Lastly, for the set of basis vectors $\{e_{x_s}\}_{s=0}^{S-1}$ in the leaves we choose the set of standard basis vectors, i.e. one-hot unit vectors, with each basis vector e_s corresponding to a random outcome. Although different choices of basis vectors would be valid as well, For the A_k we use, as already mentioned, the orthonormal discrete Fourier transform matrix, which we truncate if $B < S$.

6. Related Work

6.1. Squared Cirucuits

The work closest related to ours are the sum of compatible square circuits by Loconte et al. (2024b). As shown in Section 5.1 we recover sum of square circuits (introduced earlier as PSD circuits by Sladek et al. (2023)) if we use a polyadic decomposition in the bilinear form, cf. Equation 15. A non-polyadic decomposition would have resulted in a computation graph closer to that of *inception circuits* introduced by Wang & Van den Broeck (2024). Such a choice, however, increases the computation cost per unit in the partition circuit from quadratic (using the vector representation) to cubic (using the operator representation).

Our positive unit preserving operator circuits provide also a different perspective on constructing non-monotone circuits. While the methods described in (Sladek et al., 2023; Loconte et al., 2024c;b; Wang & Van den Broeck, 2024) regard such circuits as sum of squares, we interpret them as probabilistic events described by positive semidefinite matrices that are combined within a circuit using unit preserving bilinear forms. As such, we also establish a strong link between the circuit literature and quantum information theory.

From a practical perspective, the (sub-complete) distributions encoded with PUnCs and NoisePUnCs tackle one of the main shortcoming of sum of squares type circuits: we do not need to compute a normalization constant to ensure that event probabilities are bounded. This especially useful during training, as it means that updating the parameters of the model still guarantees the event probability to be bounded. Otherwise one would have to compute a normalization constant and differentiate through it at every iteration of the parameter updates. This potentially hinder scaling up non-monotone models as computing the normalization constant cannot be done in the vector representation (quadratic cost) but requires matrix-matrix multiplications with cubic cost. For PUnCs computing this normalization is not necessary at all and for NoisePUnC we only need to compute the normalization constant once training has finished as post-processing step.

6.2. Tensor Networks

As already pointed out by Loconte et al. (2024c;b) squared circuits and monotone circuits share many similarities with *tensor networks* (Orús, 2014; White, 1992) developed in the condensed matter physics community and have in recent years also been applied to tackle problems in supervised as well as unsupervised machine learning (Cheng et al., 2019; Han et al., 2018; Stoudenmire & Schwab, 2016). In this regard and given that tensor networks originate in the physics community, it is rather surprising that tensor networks have so far, and to the best of our knowledge, not been formulated using POVMs.

Using our POVM formulation for PUnCs is, however, not only theoretically elegant but has also practical benefits: using this formalism leads to circuits that are normalized by construction, and we can perform learning simply by optimizing the likelihood. We contrast this to the sweeping algorithms that are usually deployed in the tensor network literature, where blocks of variables are optimized one at the time while the remaining variables are held constant. It appears that this approach is inspired by the variational ansatz taken in the density-matrix renormalization group algorithm (White, 1992), which is the original algorithm for tensor networks. Alternatives to this sweeping approach have also been proposed, such as an intricate Riemannian gradient descent optimization scheme that conserves unitarity of matrices, but are rather costly to run.

We would also like to point out a theoretical result from the tensor network literature stating that picking a complex-valued parametrization instead of a real-valued one can lead to an arbitrarily large reduction in the number of parameters (Glasser et al., 2019). While this result is formulated with respect to (exact) low rank tensor decomposition and does not apply directly to the problem of learning parameters via gradient descent, we consider this observation to be a strong theoretical indicator for the superiority of complex numbers over real numbers when parametrizing probabilistic circuits. A similar argument has also been made by Gao et al. (2022) in the context of hidden Markov models.

6.3. Theoretical Studies

First theoretical results on expressive power of polynomial functions, i.e. circuits and tensor networks, were presented in the tensor network literature in the context of tensor decomposition (Glasser et al., 2019) and complex-valued hidden Markov models (Gao et al., 2022). Recently, the works of Loconte et al. (2024c;b) and Wang & Van den Broeck (2024) have studied the relationship of different circuits classes more carefully, as well. Additionally, Loconte et al. (2024b) pointed out links between tensor networks and the circuit literature and were able to generalize earlier results from the tensor network literature by Glasser et al. (2019).

Finally, we would also like to point out theoretical results n the statistical relational AI literature. Specifically, Buchman & Poole (2017b), and Kuzelka (2020) noted that using only real-valued parametrizations (including negatives (Buchman & Poole, 2017a)), i.e. monotone functions exclusively, does not allow for fully expressive models.

7. Experimental Evaluation

Experimental Setup

For our experimental evaluation we used the MNIST family of datasets. That is, the original MNIST (Deng, 2012), FashionMNIST (Xiao et al., 2017), and also EMNIST (Cohen et al., 2017). For the implementation we used PyTorch together with the Einops library (Rogozhnikov, 2022). We set up our experiments in Lightning[2] and ran them on a DGX-2 machine with V100 Nvidia cards.

We compare different methods using the *bits per dimension* metric, which is calculated from the average negative log-likelihood ($\overline{NLL}$) as follows: $bpd{=}\overline{NLL}/(\log 2{\times}D)$ ($D{=}28^2$ for MNIST datasets).

Training Positive Operator Circuits

In our experiments we estimate the density of the different MNIST datasets by maximizing a parametrized likelihood function. We construct this function using a positive vector circuit. In order to build the underlying partition circuit, we follow the approach described by Zuidberg Dos Martires (2024): we start with a grid of 28×28 pixels and merge, in an alternating fashion, rows and columns of the image. We also allow for merging three partitions into a single partition, thereby breaking the binary character of the partition trees used so far. Having three instead of two partitions being merged can easily be accommodated for by using a Hadamard product with three factors instead of two in the internal units of a PVX. Ultimately, this allows us to handle layers with an uneven number of partitions. At the leaves we encode pixel values by associating each value to one of 256 standard basis vector of $\mathbb{R}^{256}$.

We performed training by minimizing the negative log-likelihood using Adam (Kingma & Ba, 2014) with default hyperparameters, batch size of 50, over 100 epochs. The best model was selected using a $90-10$ train-validation data split where we monitored the negative log-likelihood on the validation set. We did not perform any hyperparameter search. Further details can be found in the configuration files of the experiments.

State-of-the-Art Baselines

We compare PVXsto three state-of-the-art tractable density estimators from the literature: quadrature of probabilistic circuits (QPCs) (Gala et al., 2024), hidden Chow-Liu trees (HCLTs) (Liu & Van den Broeck, 2021), and sparse HCLTs (SHCLTs) (Dang et al., 2022). All three methods are probabilistic circuits and they all use hidden Chow-Liu trees as their underlying structure. This tree is learned and dataset specific. The difference between QPCs and HCLTs is that the former is obtained by approximating a continuous latent

[2] https://lightning.ai/

variable extension of probabilistic circuits using numerical quadrature. The difference between HCLTs and SHCLTs is that the latter allows for dynamically adapting the number of components per partition during parameter learning. This means that SHCLTs can focus their computational budget on information-rich parts of the circuit.

Q1: How Do the Different PVXParametrizations Fair Against Each Other?

We construct PVXsfollowing the prescription of (Zuidberg Dos Martires, 2024), as described above. Furthermore, we use $B = 128$ components per partition. As for the parametrization we study four different variants: PVX_0^{FR}, $\text{PVX}_{2\pi}^{FR}$, PVX_0^{R1}, and $\text{PVX}_{2\pi}^{R1}$. The subscripts (0 or 2π) indicate whether we limit the phase to be zero or whether we allow the phase to be a learnable parameter. Note that having a tunable phase doubles the number of (real-valued) parameters. The superscript indicates whether the parametrization uses a full-rank (FR) density matrix in the root or a rank-one ($R1$) density matrix. As such, PVX_0^{R1} is equivalent to the squared monotonic probabilistic circuits (MPC^{2}s) used in (Loconte et al., 2024c), and $\text{PVX}_{2\pi}^{R1}$ generalizes their NPC^{2}s from the real domain to the entire complex domain. Concretely, MPC^{2}s are rank-one PVXswith the phase fixed to zero and NPC^{2}s are rank-one PVXswith the phase fixed to values in the set $\{0, \pi\}$. We do not experiment on the latter.

We report the obtained bpds for the different PVXparametrizations in the first four columns of Table 1. In general, we see that the complex-valued parametrizations are either on par or outperform the corresponding zero-phase parametrizations. The notable exception is the FashionMNIST benchmark where the two parametrizations with no phase (PVX_0^{FR} and PVX_0^{R1}) perform best. Comparing full-rank to rank-one parametrizations we see a more important impact for the zero-phase parametrization than for the complex-valued parametrization.

Overall, $\text{PVX}_{2\pi}^{FR}$ constitutes the best performing parametrization being best-in-class on all benchmarks but FashionMNIST.

Q2: How Do PVXsFair Against the State of the Art?

In order to compare PVXsto state-of-the-art circuits all methods (PVX, QPC, HCLT, SHCLT) were given a computational budget of $B = 128$ components per partition (on average for SHCLT). The results for the competing methods were taken from the respective papers – except for HCLTs. For HCLTs we took the bpds reported by Gala et al. (2024) as they achieved stronger results with their implementation compared to the originally reported performance (Liu & Van den Broeck, 2021).

We see in Table 1 that QPCs and HCLTs are in general outperformed by PVX– in particular $\text{PVX}_{2\pi}^{FR}$. The FashionMNIST benchmark constitutes again an outlier in this regard. The only methods outperforming PVXsare the SHCLTs, which is due to them being able to dynamically allocate compute budget to informative parts of the circuit. It is noteworthy that for the EMNIST-BYCLASS benchmark all four PVXparametrization exhibit strong performance with $\text{PVX}_{2\pi}^{FR}$ and $\text{PVX}_{2\pi}^{R1}$ even outcompeting SHCLTs.

Discussion

In our experimental evaluation we did not perform an explicit comparison to the method of Loconte et al. (2024c) as they were not able to find any improvements of allowing non-negative parameters in probabilistic circuit for discrete data. They stipulated that "simple categorical distribution can already capture any discrete distribution with finite support and a (subtractive) mixture thereof might not yield additional benefits". In our experimental evaluation we refute this conjecture, and show that using complex-valued parameters leads to noticeable gains when performing density estimation on discrete data. The exception being of course the FashionMNIST benchmark. We believe that this is due to our optimization method that was not tuned in any way. In this regard we believe that developing tailor-made optimization algorithms for PVXsmight further boost their performance. Furthermore, one can envisage, similar to SHCLTs, dynamically adapting the compute budget per partition, which should again improve the density estimation capacities of PVXs.

8. Conclusions & Future Work

Based on first principles from quantum information theory, we constructed positive operator circuits – a novel class of probabilistic tractable models: Their construction as partition circuits allows for layer-wise parallelization and execution on modern machine learning hardware. Using unconstrained gradient-descent we showed that POXs, and PVXsspecifically, constitute a promising addition to the zoo of tractable probabilistic models.

In future work we would like to investigate in more detail theoretical properties of POXsand how they differ from other tractable models. Ideally one would find an exponential separation between real-valued and complex-valued circuits. On the practical side it remains, however, to be seen whether such a separation has an equally important impact on real-world datasets.

This relates to another open question, that of learning in POXs. We presented a rather simple learning approach for parameters tailored towards neural architecture. It might be the case that more sophisticated techniques have to be deployed for large-scale POXs, as they might exhibit the

Table 1: Test set bpd for MNIST datasets (lower is better).

	PVX_0^{FR}	$\mathrm{PVX}_{2\pi}^{FR}$	PVX_0^{R1}	$\mathrm{PVX}_{2\pi}^{R1}$	QPC	HCLT	SHCLT
MNIST	1.16	1.16	1.24	1.17	1.18	1.21	1.14
FashionMNIST	3.37	3.55	3.44	3.55	3.27	3.34	3.27
E-MNIST	1.69	1.63	1.76	1.64	1.66	1.70	1.52
E-LETTERS	1.62	1.60	1.70	1.61	1.70	1.75	1.58
E-BALANCED	1.65	1.64	1.73	1.64	1.73	1.78	1.60
E-BYCLASS	1.47	1.47	1.56	1.49	1.67	1.73	1.54

problem of barren plateaus (Ragone et al., 2023) – a well known issue in quantum machine learning (Biamonte et al., 2017; Huggins et al., 2019). Furthermore, and as already pointed out by Loconte et al. (2024c), computing the normalization constant Z requires the evaluation of the circuit in the POXrepresentation. This gives rise to a rather expensive cubic computation cost during learning (when compared to the quadratic cost of PVXevaluations). Avoiding this issue would allow to drastically scale PVXs. In order, to scale PVXsit might also be necessary to use more sophisticated structures than binary trees with every partition having the same number of components B per partition, cf. SHCLTs.

References

Agarwal, S. and Bläser, M. Probabilistic generating circuits-demystified. In *Forty-first International Conference on Machine Learning*, 2024.

Araujo, A., Negrevergne, B., Chevaleyre, Y., and Atif, J. Understanding and training deep diagonal circulant neural networks. In *ECAI 2020*, pp. 945–952. IOS Press, 2020.

Baksalary, J. K., Pukelsheim, F., and Styan, G. P. Some properties of matrix partial orderings. *Linear Algebra and its Applications*, 119:57–85, 1989.

Biamonte, J., Wittek, P., Pancotti, N., Rebentrost, P., Wiebe, N., and Lloyd, S. Quantum machine learning. *Nature*, 549(7671):195–202, 2017.

Buchman, D. and Poole, D. Negative probabilities in probabilistic logic programs. *International Journal of Approximate Reasoning*, 83:43–59, 2017a.

Buchman, D. and Poole, D. Why rules are complex: Real-valued probabilistic logic programs are not fully expressive. In *UAI*, 2017b.

Carroll, J. D. and Chang, J.-J. Analysis of individual differences in multidimensional scaling via an n-way generalization of "eckart-young" decomposition. *Psychometrika*, 35(3):283–319, 1970.

Cheng, S., Wang, L., Xiang, T., and Zhang, P. Tree tensor networks for generative modeling. *Physical Review B*, 99 (15):155131, 2019.

Cohen, G., Afshar, S., Tapson, J., and Van Schaik, A. Emnist: Extending mnist to handwritten letters. In *2017 international joint conference on neural networks*, 2017.

Dang, M., Liu, A., and Van den Broeck, G. Sparse probabilistic circuits via pruning and growing. In *Advances in Neural Information Processing Systems*, 2022.

Darwiche, A. Decomposable negation normal form. *Journal of the ACM (JACM)*, 48(4):608–647, 2001a.

Darwiche, A. On the tractable counting of theory models and its application to truth maintenance and belief revision. *Journal of Applied Non-Classical Logics*, 11(1-2): 11–34, 2001b.

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

Darwiche, A. Sdd: A new canonical representation of propositional knowledge bases. In *Twenty-Second International Joint Conference on Artificial Intelligence*, 2011.

de Colnet, A. and Mengel, S. A compilation of succinctness results for arithmetic circuits. In *International Conference on Principles of Knowledge Representation and Reasoning, KR 2021*, 2021.

Deng, L. The mnist database of handwritten digit images for machine learning research. *IEEE signal processing magazine*, 2012.

Gala, G., de Campos, C., Peharz, R., Vergari, A., and Quaeghebeur, E. Probabilistic integral circuits. In *International Conference on Artificial Intelligence and Statistics*, 2024.

Gao, X., Anschuetz, E. R., Wang, S.-T., Cirac, J. I., and Lukin, M. D. Enhancing generative models via quantum correlations. *Physical Review X*, 12(2):021037, 2022.

Glasser, I., Sweke, R., Pancotti, N., Eisert, J., and Cirac, I. Expressive power of tensor-network factorizations for probabilistic modeling. In *Advances in neural information processing systems*, 2019.

Han, Z.-Y., Wang, J., Fan, H., Wang, L., and Zhang, P. Unsupervised generative modeling using matrix product states. *Physical Review X*, 8(3):031012, 2018.

Harviainen, J., Ramaswamy, V. P., and Koivisto, M. On inference and learning with probabilistic generating circuits. In *Uncertainty in Artificial Intelligence*, pp. 829–838. PMLR, 2023.

Huggins, W., Patil, P., Mitchell, B., Whaley, K. B., and Stoudenmire, E. M. Towards quantum machine learning with tensor networks. *Quantum Science and technology*, 4(2):024001, 2019.

Jing, L., Shen, Y., Dubcek, T., Peurifoy, J., Skirlo, S., LeCun, Y., Tegmark, M., and Soljačić, M. Tunable efficient unitary neural networks (eunn) and their application to rnns. In *International Conference on Machine Learning*, pp. 1733–1741. PMLR, 2017.

Khosravi, P., Choi, Y., Liang, Y., Vergari, A., and Van den Broeck, G. On tractable computation of expected predictions. *Advances in Neural Information Processing Systems*, 32, 2019.

Kiani, B., Balestriero, R., LeCun, Y., and Lloyd, S. projunn: efficient method for training deep networks with unitary matrices. *Advances in Neural Information Processing Systems*, 35:14448–14463, 2022.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

Kuzelka, O. Complex markov logic networks: Expressivity and liftability. In *Conference on Uncertainty in Artificial Intelligence*, pp. 729–738. PMLR, 2020.

Lezcano-Casado, M. and Martınez-Rubio, D. Cheap orthogonal constraints in neural networks: A simple parametrization of the orthogonal and unitary group. In *International Conference on Machine Learning*, pp. 3794–3803. PMLR, 2019.

Liu, A. and Van den Broeck, G. Tractable regularization of probabilistic circuits. In *Advances in Neural Information Processing Systems*, 2021.

Loconte, L., Mari, A., Gala, G., Peharz, R., de Campos, C., Quaeghebeur, E., Vessio, G., and Vergari, A. What is the relationship between tensor factorizations and circuits (and how can we exploit it)? *arXiv preprint arXiv:2409.07953*, 2024a.

Loconte, L., Mengel, S., and Vergari, A. Sum of squares circuits. *arXiv preprint arXiv:2408.11778*, 2024b.

Loconte, L., Sladek, A. M., Mengel, S., Trapp, M., Solin, A., Gillis, N., and Vergari, A. Subtractive mixture models via squaring: Representation and learning. In *Proceedings of the International Conference on Learning Representations*, 2024c.

Martens, J. and Medabalimi, V. On the expressive efficiency of sum product networks. *arXiv preprint arXiv:1411.7717*, 2014.

Mhammedi, Z., Hellicar, A., Rahman, A., and Bailey, J. Efficient orthogonal parametrisation of recurrent neural networks using householder reflections. In *International Conference on Machine Learning*, pp. 2401–2409. PMLR, 2017.

Nielsen, M. A. and Chuang, I. L. Quantum computation and quantum information. *Phys. Today*, 54(2):60, 2001.

Orús, R. A practical introduction to tensor networks: Matrix product states and projected entangled pair states. *Annals of physics*, 349:117–158, 2014.

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Shao, X., Trapp, M., Kersting, K., and Ghahramani, Z. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Uncertainty in Artificial Intelligence*, 2019.

Peharz, R., Lang, S., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Van den Broeck, G., Kersting, K., and Ghahramani, Z. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *International Conference on Machine Learning*, 2020.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops*. IEEE, 2011.

Ragone, M., Bakalov, B. N., Sauvage, F., Kemper, A. F., Marrero, C. O., Larocca, M., and Cerezo, M. A unified theory of barren plateaus for deep parametrized quantum circuits. *arXiv preprint arXiv:2309.09342*, 2023.

Rogozhnikov, A. Einops: Clear and reliable tensor manipulations with einstein-like notation. In *International Conference on Learning Representations*, 2022.

Schur, I. Bemerkungen zur Theorie der beschränkten Bilinearformen mit unendlich vielen Veränderlichen. 1911.

Shpilka, A. and Yehudayoff, A. Arithmetic circuits: A survey of recent results and open questions. *Foundations and Trends® in Theoretical Computer Science*, 5(3–4): 207–388, 2010.

Sladek, A. M., Trapp, M., and Solin, A. Encoding negative dependencies in probabilistic circuits. In *The 6th Workshop on Tractable Probabilistic Modeling*, 2023.

Stoudenmire, E. and Schwab, D. J. Supervised learning with tensor networks. *Advances in neural information processing systems*, 2016.

Valiant, L. G. Negation can be exponentially powerful. In *Proceedings of the eleventh annual ACM symposium on theory of computing*, pp. 189–196, 1979.

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

von Neumann, J. Wahrscheinlichkeitstheoretischer Aufbau der Quantenmechanik. *Nachrichten von der Gesellschaft der Wissenschaften zu Göttingen, Mathematisch-Physikalische Klasse*, 1927:245–272, 1927.

Wang, B. and Van den Broeck, G. On the relationship between monotone and squared probabilistic circuits. In *The 7th Workshop on Tractable Probabilistic Modeling*, 2024.

White, S. R. Density matrix formulation for quantum renormalization groups. *Physical review letters*, 69(19):2863, 1992.

Wisdom, S., Powers, T., Hershey, J., Le Roux, J., and Atlas, L. Full-capacity unitary recurrent neural networks. *Advances in neural information processing systems*, 29, 2016.

Wiseman, H. M. and Milburn, G. J. *Quantum measurement and control*. Cambridge university press, 2009.

Xiao, H., Rasul, K., and Vollgraf, R. Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms. *arXiv preprint arXiv:1708.07747*, 2017.

Zhang, H., Holtzen, S., and Broeck, G. On the relationship between probabilistic circuits and determinantal point processes. In *Conference on Uncertainty in Artificial Intelligence*, pp. 1188–1197. PMLR, 2020.

Zuidberg Dos Martires, P. Probabilistic neural circuits. In *Proceedings of the AAAI Conference on Artificial Intelligence*, 2024.

Appendix

A. Proofs

A.1. Proof of Proposition 2.5

Proposition 2.5. *The expression in Equation 2 defines a valid probability distribution.*

Proof. While this is a well-known result we were not able to identify a concise proof in the literature and provide therefore, for the sake of completeness here. To this end, we first show that $p(i) \geq 1$, for each i.

$$\begin{aligned} p(i) &= \mathrm{Tr}[E(i)\rho] = \mathrm{Tr}[D(i)D^*(i)\rho] \\ &= \mathrm{Tr}[D(i)^*\rho D(i)]. \end{aligned} \tag{20}$$

Here we used the fact that $E(i)$ is PSD and factorized it into the product $D(i)D^*(i)$. Then we used the fact that the trace is invariant under cyclical shifts. Clearly, the matrix $D(i)^*\rho D(i)$ is PSD as we have for every vector x:

$$x^* D(i)^*\rho D(i)x = y^*\rho y \geq 0 \quad \text{with } y = Dx. \tag{21}$$

As the trace of a PSD matrix is positive we have that $p(i)$ is a positive real number, i.e. $p(i){\geq}0$ for every i.

Secondly, we show that $p(i)$ is normalized:

$$\sum_{i=1}^{N} p(i) = \sum_{i=1}^{N} \mathrm{Tr}[E(i)\rho] = \mathrm{Tr}\left[\sum_{i=1}^{N} E(i)\rho\right] = \mathrm{Tr}[\rho], \tag{22}$$

where we used Equation 1. Exploiting the fact that the trace of a density matrix is one gives us indeed $\sum_{i=1}^{N} p(i) = 1$, and we can conclude that $p(i)$ is a valid probability distribution. □

A.2. Proof of Proposition 3.4

Proposition 3.4. *Positive operator circuits are PSD.*

Proof. This can be seen by first inspecting the leaves where we have $A_k \times e_{x_k} \otimes e^*_{x_k} \times A^*_k$. Using a similar argument as made in Equation 21 we know that all the leaves carry PSD matrices. Passing these on recursively to the bilinear forms in the internal units retains the positive semi-defiteness (imposed by definition). We conclude that computation units in of positive operator circuit output PSD matrices. □

A.3. Proof of Proposition 3.6

Proposition 3.6. *Let $\mathbf{X}$ denote a set of random variables with sample space $\Omega(\mathbf{X})$. Then the set $\{\widehat{O}(\mathbf{x})\}_{\mathbf{x}\in\Omega(\mathbf{X})}$ of positive unitary circuis forms a POVM.*

Proof. Given that positive unitary circuits are by definition positive operator circuits we already have that:

$$\forall \mathbf{x} \in \Omega(\mathbf{X}) : \widehat{O}(\mathbf{x}) \text{ is PSD.} \tag{23}$$

Next we show that $\sum_{\mathbf{x}\in\Omega(\mathbf{X})} \widehat{O}(\mathbf{x}) = \mathbb{1}$. Here we observe that in the computation units we have:

$$\begin{aligned} \sum_{\mathbf{x}_k\in\Omega(\mathbf{X}_k)} \widehat{O}_k(\mathbf{x}_k) &= \sum_{\mathbf{x}_{k_l}}\sum_{\mathbf{x}_{k_r}} \mathscr{B}(\widehat{O}_k(\mathbf{x}_{k_l}), \widehat{O}_k(\mathbf{x}_{k_r})) \\ &= \mathscr{B}\left(\sum_{\mathbf{x}_{k_l}} \widehat{O}_k(\mathbf{x}_{k_l}), \sum_{\mathbf{x}_{k_r}} \widehat{O}_k(\mathbf{x}_{k_r})\right) \end{aligned} \tag{24}$$

This lets us push down the summation of a specific variable to the corresponding leaf where the variable is given as input, where we then have:

$$\begin{aligned}\sum_{\mathbf{x}_k \in \Omega(\mathbf{X}_k)} \widehat{O}_k(\mathbf{x}_k) &= \sum_{x_k \in \Omega(X_k)} A_k e_{x_k} \otimes e^*_{x_k} A^*_k \\ &= A_k \left(\sum_{x_k \in \Omega(X_k)} e_{x_k} \otimes e^*_{x_k} \right) A^*_k \\ &= A_k \mathbb{1}_{\Omega(\mathbb{X}_\mathbb{k})} A^*_k = \mathbb{1}_\mathbb{k} \end{aligned} \tag{25}$$

We now exploit that the bilinear forms in a positive unitary circuit are unit preserving (cf. Defintion 3.5), which gives us indeed $\sum_{\mathbf{x} \in \Omega(\mathbf{X})} \widehat{O}(\mathbf{x}) = \mathbb{1}$. □

A.4. Proof of Proposition 4.3

Proposition 4.3. *Noisy POVMs induce a sub-complete probability distributions.*

Proof. Let ρ be a density matrix and $\{E(i)\}_{i=0}^{I-1}$ be a POVM such that $\sum_{i=0}^{I-1} E(i) = M$. We then have

$$\mathrm{Tr}[(\mathbb{1} - M)\rho] \geq 0, \tag{26}$$

holds as $\mathbb{1} - M$ is PSD by definition. Pushing the trace over the subtraction and rearranging terms yields:

$$\mathrm{Tr}[\mathbb{1}\rho] - \mathrm{Tr}[M\rho] \geq 0 \Leftrightarrow \mathrm{Tr}[M\rho] \leq 1. \tag{27}$$

This means that the induced probability distribution $p(i) = \mathrm{Tr}[E(i)\rho]$ is indeed sub-complete. □

A.5. Proof of Proposition 4.5

Proposition 4.5. *NoisePUnCs induce sub-complete probability distributions.*

Proof. Let $\mathbf{X}$ be a set of random variables and $Q(\mathbf{x}) = q(\mathbf{x})\widehat{O}(\mathbf{x})$ with $\mathbf{x} \in \Omega(\mathbf{X})$ be a NoisePUnC inducing a probability distribution

$$\pi_\mathbf{X}(\mathbf{x}) = \mathrm{Tr}\Big[q(\mathbf{x})\widehat{O}(\mathbf{x})\rho\Big]. \tag{28}$$

As $q(\mathbf{x})$ is a scalar we can write

$$\pi_\mathbf{X}(\mathbf{x}) = q(\mathbf{x})\,\mathrm{Tr}\Big[\widehat{O}(\mathbf{x})\rho\Big] = q(\mathbf{x})p_X(\mathbf{x}), \tag{29}$$

where $p_X(\mathbf{x})$ is the (complete) probability distribution induced by the set $\{\widehat{O}(\mathbf{x})\}_{\mathbf{x}\in\Omega(\mathbf{X})}$. As $0 \leq q(\mathbf{x}) \leq 1$ (by definition) and $0 \leq p_\mathbf{X}(\mathbf{x}) \leq 1$ we have also that $0 \leq \pi_\mathbf{X}(\mathbf{x}) \leq 1$. This shows that each event $\mathbf{x} \in \Omega(\mathbf{X})$ has a positive probability.

For the sub-completeness we observe that the sum of positive terms

$$\sum_{\mathbf{x}\in\Omega(\mathbf{X})} p_\mathbf{X}(\mathbf{x}) = 1. \tag{30}$$

If we weigh each term in the sum with a scalar $q(\mathbf{x}) \in [0,1]$ we immediately conclude that $\sum_{\mathbf{x}\in\Omega(\mathbf{X})} \pi_\mathbf{X}(\mathbf{x}) \leq$ and that we have indeed a sub-complete probability distribution. □

A.6. Proof of Proposition 4.7

Proposition 4.7. *A soft counting circuit $q(\mathbf{x})$ satisfies* $0 \leq q(\mathbf{x}) \leq 1$.

Proof. We consider the two extreme cases where on teh one hand the W_k in the leaves have only zeros as entries and on the other hand only ones. For the first case it is easy to see that multiplying any vector one_hot(x_k) with a zero matrix results in a zero vector $q_k(x_k)$. Evaluting the circuit with these zero vector from the leaves trivially gives us $p_{root}(\mathbf{x}_{root}) = 0$.

For the second case we observe that multiplying any vector one_hot(x_k) with a matrix having only ones as entries results in a vector $q_k(x_k)$ having only ones as entries. As the matrices W_k in the internal nodes and the root node are row-normalized this is preserved throughout the circuit and we have $p_{root}(\mathbf{x}_{root}) = 1$.

Finally, as soft counting circuits are monotone functions all other cases fall between these two extremes and we conclude that in general $0 \leq q(\mathbf{x}) \leq 1$. □

A.7. Proof of Proposition 5.1

Proposition 5.1. *Let N be the number of variables $\mathbf{x}_k$, having all the domain size S, and let the L_k and R_k matrices be $B \times B$ square matrices, with $B \leq S$. We can perform marginal inference in PUnCs in $\mathcal{O}(NS^3)$ when using the bilinear form from Equation 15.*

Proof. We determine the computational cost of evaluating the circuit by determining the cost of the individual computation units. We start at the leaves where we first have a Kronecker product $e_{x_k} \otimes e^*_{x_k}$, which can be performed in $\mathcal{O}(S^2)$. The matrix products with A_k and A^*_k can then be performed in $\mathcal{O}(B^2 S)$, which implies $\mathcal{O}(\mathcal{O}(S^2))$ if $B \leq N$. In the bilinear form (cf. Equation 15) we perform four matrix-matrix product in $\mathcal{O}(B^3)$. This is followed by a Hadamard product in $\mathcal{O}(B^2)$, This means that we have for the individual computation units in the partition a circuit a cost of $\mathcal{O}(S^3)$.

It is also easy to show that for binary tree partition circuits we have a total of $\mathcal{O}(N)$ computation units. This results in a computational complexity of $\mathcal{O}(NS^3)$.

We note the computation of trace of the matrix-matrix multiplication between $\widehat{O}_{root}(\mathbf{x})$ and the density matrix ρ necessary for obtaining a probability can be done in $\mathcal{O}(B^2)$.

Given that we can evaluate PUnCs $\mathcal{O}(NS^3)$ and also that we can marginalize out variables by pushing the summation to the leaves (cf. the proof of Proposition 3.6), we can conclude that marginal inference has complexity $\mathcal{O}(NS^3)$, as well. □

A.8. Proof of Proposition 5.3

Lemma A.1. *For the circuits defined in Definition 3.2 and Definition 5.2 it holds that*

$$\forall k : O_k(\mathbf{x}_k) = v_k(\mathbf{x}_k) \otimes v^*_k(\mathbf{x}_k). \tag{31}$$

Proof. We start the proof at the leave where we have

$$O_k(x_k) = A_k e_{x_k} \otimes e^*_{x_k} A^*_k \tag{32}$$

and Equation 31 holds by definition. In the computation untis into which the leaves feed, we then have

$$\begin{aligned} O_k &= L_k O_{k_l} L^*_k \odot R_k O_{k_r} R^*_k \\ &= L_k (v_{k_l} \otimes v^*_{k_l}) L^*_k \odot R_k (v_{k_r} \otimes v^*_{k_r}) R^*_k, \end{aligned} \tag{33}$$

where we omited the explicit dependencies on the variables x_k, x_{k_l}, and x_{k_r}. Next we use the mixed-product property of the Hadamard product and Kronecker product to get

$$\begin{aligned} O_k &= L_k v_{k_l} \odot R_k v_{k_r} \otimes v^*_{k_l} L^*_k \odot v^*_{k_r} R^*_k \\ &= v_k \otimes v^*_k. \end{aligned} \tag{34}$$

Repeating this argument recursively until the root of the circuit concludes the proof. □

Proposition 5.3. *The expression $\|\gamma \times v_{root}(\mathbf{x})\|^2$, with $\rho = \gamma \times \gamma^*$ computes the same probability as* $\mathrm{Tr}[O_{root}(\mathbf{x})\rho]$.

Proof. The proof starts by simply plugging in the vector representation of O_{root} into the expression $\mathrm{Tr}[O_{\text{root}}\rho]$ and rather straightforward.

$$\begin{aligned} \mathrm{Tr}[O_{\text{root}}\rho] &= \mathrm{Tr}\left[\left(v_{\text{root}} \otimes v^*_{\text{root}}\right) \times \rho\right] \\ &= v^*_{\text{root}} \times \rho \times v_{\text{root}} \\ &= v^*_{\text{root}} \times \gamma^* \times \gamma \times v_{\text{root}} = \|\gamma \times v_{\text{root}}\|^2 \end{aligned} \tag{35}$$

□

A.9. Proof of Proposition 5.4

Proposition 5.4. *PUnCs in their vector representation compute joint probabilities in time* $\mathcal{O}(NS^2)$

Proof. Following a similar train of thought as the proof in Section A.7 we can derive the computational cost of evaluating a positive vector circuit to be $\mathcal{O}(NBS)$ implying $\mathcal{O}(NS^2)$ when $B \leq S$. The major difference between matrix representation of operator circuits and their vector representation is that for the former the internal computation units output matrices ($O_k(\mathbf{x}_k) \in \mathbb{C}^{B\times B}$) while for the latter vectors are outputted ($v_k(\mathbf{x}_k) \in \mathbb{C}^B$).

As $v_{root}(X_{rooot})$ is computable in $\mathcal{O}(NBS)$ and $\|\gamma \times v_{\text{root}}\|^2$ is computable in $\mathcal{O}(B^2)$, we can conclude that the probability $p_{\mathbf{X}}(\mathbf{x})$ can be computed in $\mathcal{O}(NS^2)$. □

B. Computing the Normalization for NoisePUnCs

As we dscribe in Equation 13 multiplying a soft counting circuit $q(\mathbf{x})$ with a PUnC $\widehat{O}(\mathbf{x})$ gives us in the internal nodes a mixture of operator:

$$Q_{ki}(\mathbf{x}) = \sum_j w_{kij} \widetilde{Q}_{kj}, \tag{36}$$

with $\widetilde{Q}_{kj} = \mathscr{B}_k(q_{k_l j}\widehat{O}_{k_l}, q_{k_r j}\widehat{O}_{k_r})$. For the specific case of polyadic decompositions in the bilinear form we get:

$$\widetilde{Q}_{kj} = \left(D_k^{1/2} U_k \times q_{k_l j}\widehat{O}_{k_l} \times U_k^* D_k^{1/2}\right) \odot \left(D_k^{-1/2} V_k \times q_{k_r j}\widehat{O}_{k_r} \times V_k^* D_k^{-1/2}\right). \tag{37}$$

If we now apply Equation 13 recursively, we end up in the leaves. Here we first write the matrix vector multiplications in the leaves of the soft counting circuit element-wise:

$$\begin{aligned} & q_k(x_k) = W_k \times \text{one\_hot}(x_k) \\ \Leftrightarrow\ & q_{ki}(x_k) = \sum_j w_{kij} [\![x_k = j]\!] \end{aligned} \tag{38}$$

where we write the dependency on the x_k variable again explicitly. This gives us for the leaves of a NoisePUnC:

$$\begin{aligned} Q_{ki}(x_k) &= q_{ki}(x_k)\widehat{O}_k(x_k), \\ &= \left(\sum_j w_{kij} [\![x_k = j]\!]\right) A_k \left(e_{x_k} \otimes e_{x_k}^*\right) A_k^* \\ &= w_{kix_k} A_k \left(e_{x_k} \otimes e_{x_k}^*\right) A_k^* \end{aligned} \tag{39}$$

As taking the product of two circuit with the identical partition tree results again in a circuit with the same partition tree, we can again push the summation over the variables $\mathbf{x} \in \Omega(\mathbf{X})$ to the respective leaves, analogous to the proof in Appendix A.3.

$$\begin{aligned} \sum_{x_k \in \Omega(X_k)} Q_{ki}(x_k) = \sum_{x_k \in \Omega(X_k)} q_{ki}(x_k)\widehat{O}_k(x_k) &= \sum_{x_k \in \Omega(X_k)} w_{kix_k} A_k \left(e_{x_k} \otimes e_{x_k}^*\right) A_k^* \\ &= A_k \left(\sum_{x_k \in \Omega(X_k)} w_{kix_k} \left(e_{x_k} \otimes e_{x_k}^*\right)\right) A_k^* \\ &= A_k \times \begin{pmatrix} w_{ki0} & 0 & \cdots & 0 \\ 0 & w_{ki1} & \cdots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \cdots & w_{ki(S-1)} \end{pmatrix} \times A_k^* \\ &= A_k \times \text{diag}(w_{ki}) \times A_k^* \end{aligned} \tag{40}$$

We can now also see that when we have in the leaves that $W_{kij} = 1$ for every k, i, and j that we recover the (non-noisy) PUnCs as a special case.

For computing the normalization constant of a NoisePUnC this means that we first compute in the leaves the prouct of $\operatorname{diag}(w_{ki})$ sandwidched between A_k and A_k^*. For each of the leaves we have C of these products. Hence, the computation of a single leaf can be done in $\mathcal{O}(CB^2S)$. Using agina that $B \leq S$ we get $\mathcal{O}(CS^3)$.

In the computation units we then compute C times a bilinear form giving us $\mathcal{O}(CS^3)$. This is followed by the computation of a mixture of operators for which we need $\mathcal{O}(C^2S^2)$. Given that there are $\mathcal{O}(N)$ computation units we get that the computation of the normalization constant takes $\big(NCS^3 + C^2S^2\big)$. The advantage of NoisePUnCs over other circuit architectures (Loconte et al., 2024b; Wang & Van den Broeck, 2024) is that this normalization constant had to be computed once training has finished and does not need to be differentiated through.