Sum of Squares Circuits

Lorenzo Loconte[1], **Stefan Mengel**[2], **Antonio Vergari**[1]
[1]School of Informatics, University of Edinburgh, UK
[2]University of Artois, CNRS, Centre de Recherche en Informatique de Lens (CRIL), France
l.loconte@sms.ed.ac.uk, mengel@cril-lab.fr, avergari@ed.ac.uk

Abstract

Designing expressive generative models that support exact and efficient inference is a core question in probabilistic ML. Probabilistic circuits (PCs) offer a framework where this tractability-vs-expressiveness trade-off can be analyzed theoretically. Recently, *squared* PCs encoding subtractive mixtures via negative parameters have emerged as tractable models that can be exponentially more expressive than *monotonic* PCs, i.e., PCs with positive parameters only. In this paper, we provide a more precise theoretical characterization of the expressiveness relationships among these models. First, we prove that squared PCs can be less expressive than monotonic ones. Second, we formalize a novel class of PCs – *sum of squares PCs* – that can be exponentially more expressive than *both* squared and monotonic PCs. Around sum of squares PCs, we build an expressiveness hierarchy that allows us to precisely *unify and separate* different tractable model classes such as Born Machines and PSD models, and other recently introduced tractable probabilistic models by using complex parameters. Finally, we empirically show the effectiveness of sum of squares circuits in performing distribution estimation.

Code — https://github.com/april-tools/sos-npcs
Extended version — https://arxiv.org/abs/2408.11778

1 Introduction

We design and learn *expressive* probabilistic models to compactly represent the complex distribution we assume has generated the data we deal with. At the same time, a key requirement to effectively reason about such a distribution is that we can perform inference exactly and efficiently, i.e., *tractably*. This is especially relevant in safety-critical real-world applications where reliable inference is required (Ahmed et al. 2022; Marconato et al. 2024b,a).

Quantifying this trade-off between tractability and expressiveness can be done within the framework of probabilistic circuits (PCs) (Vergari, Di Mauro, and Van den Broeck 2019), which are deep computational graphs that generalize many tractable representations in ML and AI (Choi, Vergari, and Van den Broeck 2020). For example, within the framework of PCs, guaranteeing the tractability of certain inference scenarios, e.g., marginals, MAP inference, or computing divergences, directly translates into ensuring that these computational graphs have certain structural properties (Vergari et al. 2021). Expressiveness, on the other hand, can be theoretically characterized in terms of the circuit size (and hence number of learnable parameters) (Shpilka and Yehudayoff 2010). Furthermore, one can precisely characterize that two model classes can capture the same probability distributions, if we can *reduce* one to the other in polytime. Conversely, to *separate* two model classes in terms of expressiveness, one has to prove that a function can be exactly captured by a circuit class but not by the other, unless this one has exponential size (de Colnet and Mengel 2021).

To ensure that a circuit outputs non-negative values only – the minimum requirement to model a valid density – PCs are classically represented and learned with non-negative parameters only, i.e., they are *monotonic* (Shpilka and Yehudayoff 2010). This prominent circuit class can only represent mixture models whose component densities are added. To fill this gap, Loconte et al. (2024a) introduced a special class of *non-monotonic* PCs, i.e., circuits with negative parameters, thus allowing to subtract mixture components as well. Non-negativity of the circuit outputs is ensured by tractably squaring the PCs (Vergari et al. 2021). Loconte et al. (2024a) proved an exponential separation between monotonic and non-monotonic squared circuits, showing that the latter can be more expressive than the former. However, one question remains open: *can squared circuits compactly compute all functions compactly encoded by monotonic circuits?*

In this paper, we provide a negative answer, by proving another separation that goes in the opposite direction: certain distributions that can be captured by polynomially-sized monotonic circuits require a squared circuit of exponential size. To overcome this limitation, we introduce a larger model class that can be more expressive than both: *sum of squares* PCs. While sum of squares forms are a staple in modeling non-negative polynomials (Marshall 2008), understanding when and how they lead to compact circuit representations is still not well understood. We close this gap by building a fine-grained expressiveness hierarchy around the class of squared PCs and their sums as illustrated in Fig. 1. We not only precisely characterize how other tractable models such as PSD models (Rudi and Ciliberto 2021), Born Machines (Glasser et al. 2019), squared neural families (Tsuchida, Ong, and Sejdinovic 2023), and Inception PCs (Wang and Van den Broeck 2024) belong to

CLASS $\mathcal{C}$	DESCRIPTION
$\pm_{\mathsf{sd}}$	Structured-decomposable circuits
$+_{\mathsf{sd}}$	Structured-decomposable monotonic circuits
$\pm^2_{\mathbb{R}}$	Squared circuits (Loconte et al. 2024a)
$\pm^2_{\mathbb{C}}$	Squared circuits with complex parameters
psd	PSD circuits (Sladek, Trapp, and Solin 2023)
Σ^2_{cmp}	Sum of compatible squares (Definition 4)
$\Delta\Sigma^2_{\mathsf{cmp}}$	Difference of two circuits in Σ^2_{cmp}

Table 1: The notation we use to refer to classes of circuits.

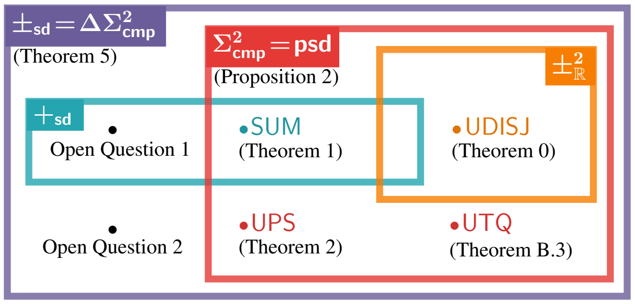

Figure 1: For each circuit class $\mathcal{C}$ (see Table 1 for a description) we illustrate with a rectangle the set of functions that can be efficiently computed by a circuit in $\mathcal{C}$. The SUM, UPS, and UTQ functions are introduced in this paper to show exponential separation results between classes $+_{\mathsf{sd}}$, $\pm^2_{\mathbb{R}}$ and Σ^2_{cmp}. We show the overlapping of classes in terms of expressiveness (denoted in the figure with $=$), and report some open questions about this hierarchy in Section 6.

this novel class of non-monotonic circuits, but also break these newly introduced boundaries by analyzing when certain PCs cannot be encoded as sum of squares. Finally, we settle the question on what is the role of complex parameters w.r.t. model expressiveness, by showing that squared complex circuits are sum of squares.

Our theoretical contributions are summarized in Fig. 1. We first prove that monotonic PCs can be exponentially more expressive than squared PCs (Section 3). Then, we introduce our sum of squares circuit class and provide two constructions to prove it can be exponentially more expressive than both monotonic and squared PCs (Section 4). We precisely relate this new class to other tractable formalisms in Section 5, and further extend it in Section 6. Finally, we empirically validate the increased expressiveness of sum of squares circuits for distribution estimation, showing they can scale to real-world data when tensorized (Section 7).

2 From Circuits to Squared PCs

We denote sets of variables in bold uppercase, e.g., $\mathbf{X} = \{X_1, \ldots, X_d\}$ while $\mathbf{x} \in \mathsf{dom}(\mathbf{X})$ denotes an assignment from the domain $\mathsf{dom}(\mathbf{X})$. We use $[n]$ for the set $\{1, \ldots, n\}$. We start by defining circuits and their properties.

Definition 1 (Circuit (Vergari et al. 2021))**.** A *circuit* c is a parameterized computational graph over $\mathbf{X}$ encoding a function $c(\mathbf{X})$ and comprising three kinds of units: *input*, *product*, and *sum*. Each product or sum unit n receives the outputs of other units as inputs, denoted as the set $\mathsf{in}(n)$. Each unit n encodes a function c_n defined as: (i) $f_n(\mathsf{sc}(n))$ if n is an input unit having *scope* $\mathsf{sc}(n) = \{Y\}$, $Y \in \mathbf{X}$, where f_n is a function defined over Y; (ii) $\prod_{j\in\mathsf{in}(n)} c_j(\mathsf{sc}(j))$ if n is a product unit; and (iii) $\sum_{j\in\mathsf{in}(n)} \theta_{n,j} c_j(\mathsf{sc}(j))$ if n is a sum unit, where each $\theta_{n,j} \in \mathbb{R}$ is a parameter of n. The scope of a non-input unit n is the union of the scopes of its inputs, i.e., $\mathsf{sc}(n) = \bigcup_{j\in\mathsf{in}(n)} \mathsf{sc}(j)$.

W.l.o.g., we assume product units to have at most two inputs. For a circuit c, this can be enforced with just a quadratic increase in its *size*, i.e., the number of edges in it, denoted as $|c|$ (Martens and Medabalimi 2014). A *probabilistic circuit* (PC) is a circuit c encoding a non-negative function, i.e., $c(\mathbf{x}) \geq 0$ for any $\mathbf{x}$, thus encoding a (possibly unnormalized) probability distribution $p(\mathbf{x}) \propto c(\mathbf{x})$. A PC c supports the tractable marginalization of any variable subset in time $\mathcal{O}(|c|)$ (Choi, Vergari, and Van den Broeck 2020) if (i) its input functions f_n can be integrated efficiently and (ii) it is *smooth* and *decomposable*, as defined next.

Definition 2 (Smoothness and decomposability (Darwiche and Marquis 2002))**.** A circuit is *smooth* if for every sum unit n, all its input units depend on the same variables, i.e, $\forall i,j \in \mathsf{in}(n)\colon \mathsf{sc}(i) = \mathsf{sc}(j)$. A circuit is *decomposable* if the inputs of every product unit n depend on disjoint sets of variables, i.e, $\forall i,j \in \mathsf{in}(n)\ i \neq j\colon \mathsf{sc}(i) \cap \mathsf{sc}(j) = \varnothing$.

Expressive efficiency also called *succinctness*, or *representational power* refers to the ability of a circuit class to encode a (possibly unnormalized) distribution in a polysize computational graph, i.e., whose size grows polynomially w.r.t. its input size. This is in contrast with "expressiveness" intended as *universal representation*: all PCs with Boolean (resp. Gaussian) inputs can express any Boolean (resp. continuous) distribution arbitrarily well, but in general at the cost of an exponential size. When we say that one class is exponentially more efficient or succinct than another we imply there exist a distribution, also called *separating function*, that cannot be captured by any polysize circuit belonging to the second class. E.g., circuits satisfying fewer or different structural properties (e.g., by relaxing decomposability in Definition 2) can be exponentially more expressive efficient than other circuit classes, i.e., they might tractably compute a larger class of functions (Martens and Medabalimi 2014).

Monotonic vs non-monotonic PCs. Designing and learning a circuit to be a PC is typically done by assuming that both the parameters and the input functions are non-negative, resulting in a *monotonic* PC (Shpilka and Yehudayoff 2010). PCs relaxing this assumption – i.e., *non-monotonic* PCs – have been shown to be more expressive efficient than monotonic ones (Valiant 1979). However, building them in a general and flexible way is challenging (Dennis 2016).

Squared PCs. Loconte et al. (2024a) showed one can efficiently represent and learn a large class of non-monotonic PCs by *squaring* circuits. Formally, given a non-monotonic circuit c whose output can also be negative, a squared PC c^2 can be computed by multiplying c with itself, i.e., $c^2(\mathbf{x}) =$

$c(\mathbf{x}) \cdot c(\mathbf{x})$. Computing the product of two circuits, can be done efficiently if the two circuits are *compatible*.

Definition 3 (Compatibility (Vergari et al. 2021))**.** Two smooth and decomposable circuits c_1, c_2 over variables $\mathbf{X}$ are *compatible* if (i) the product of any pair f_n, f_m of input functions respectively in c_1, c_2 that have the same scope can be efficiently integrated, and (ii) any pair n, m of product units respectively in c_1, c_2 that have the same scope decompose their scope over their inputs in the same way.

Multiplying two compatible circuits c_1, c_2 can be done via the $\mathsf{Multiply}$ algorithm in time $\mathcal{O}(|c_1||c_2|)$ as described in Vergari et al. (2021) and which we report in Appendix A.1. To tractably square a circuit, this needs to be compatible with itself, i.e., *structured-decomposable*. Another way to understand structured-decomposability is through the notion of the hierarchical partitioning of the circuit scope, that is induced by product units, also called *region graph* (Mari, Vessio, and Vergari 2023). For a structured-decomposable PC with univariate inputs this partitioning forms a tree, sometimes called vtree (Pipatsrisawat and Darwiche 2008) or pseudo-tree (Dechter and Mateescu 2007).

For a smooth and structured-decomposable c, the output of $\mathsf{Multiply}(c, c)$ will be a PC c^2 of size $\mathcal{O}(|c|^2)$ that is again smooth and structured-decomposable. Therefore, it can be renormalized to compute $p(\mathbf{x}) = c^2(\mathbf{x})/Z$, where $Z = \int_{\mathsf{dom}(\mathbf{X})} c^2(\mathbf{x})\mathrm{d}\mathbf{x}$ is its partition function, if it has tractable input units, e.g., indicator functions, splines or exponential families (Loconte et al. 2024a). We call PCs that are constructed as $\mathsf{Multiply}(c, c)$, for a circuit c with (potentially negative) real weights, *squared PCs* and denote by $\pm^2_{\mathbb{R}}$ the class of all such circuits. Remark that the $\mathsf{Multiply}$ algorithm has the property that its output $\mathsf{Multiply}(c, c)$ can never be smaller than its input c, so to lower bound the size of a squared PC c^2, it suffices to bound the circuit c from which it was constructed. It has been shown that there are non-negative functions that can be efficiently represented by squared PCs but not by structured monotonic PCs.

Theorem 0 (Loconte et al. (2024a))**.** There is a class of non-negative functions $\mathcal{F}$ over d variables $\mathbf{X}$ that can be represented as a PC $c^2 \in \pm^2_{\mathbb{R}}$ with size $|c^2| \in \mathcal{O}(d^2)$. However, the smallest monotonic and structured PC computing any $F \in \mathcal{F}$ has at least size $2^{\Omega(d)}$.

The class of separating functions used to show Theorem 0 consists of *uniqueness disjointness functions* (UDISJ) (De Wolf 2003) defined as embedded on a graph $G = (V, E)$, where V denotes its vertices and E its edges, as

$$\mathsf{UDISJ}(\mathbf{X}) = \left(1 - \sum\nolimits_{uv \in E} X_u X_v\right)^2 \tag{1}$$

where $\mathbf{X} = \{X_v \mid v \in V\}$ are Boolean variables.

This expressive efficiency result theoretically justifies the increased expressiveness of not only squared PCs, but also other probabilistic models that can be exactly reduced to squared PCs as shown in Loconte et al. (2024a). These include matrix-product-states (MPSs) (Pérez-García et al. 2007) – also called tensor-trains (Oseledets 2011) – which factorize a (possibly negative) function $\psi(\mathbf{X})$ over variables $\mathbf{X} = \{X_1, \ldots, X_d\}$ with $\mathsf{dom}(X_i) = [v]$ as

$$\psi(\mathbf{x}) = \sum\nolimits_{i_1=1}^{r} \sum\nolimits_{i_2=1}^{r} \cdots \sum\nolimits_{i_{d-1}=1}^{r} \mathbf{A}_1[x_1, i_1] \tag{2}$$
$$\cdot \mathbf{A}_2[x_2, i_1, i_2] \cdots \mathbf{A}_{d-1}[x_{d-1}, i_{d-2}, i_{d-1}] \mathbf{A}_d[x_d, i_{d-1}]$$

where r is the rank of the factorization, $\mathbf{A}_1, \mathbf{A}_d \in \mathbb{R}^{v \times r}$, $\mathbf{A}_j \in \mathbb{R}^{v \times r \times r}$ for all $j \in \{2, \ldots, d-1\}$, and square brackets denote tensor indexing. When ψ factorizes as in Eq. (2), it is possible to construct a structured-decomposable circuit c computing ψ of size $|c| \in \mathcal{O}(\mathrm{poly}(d, r, v))$ (Loconte et al. 2024a). To encode a distribution $p(\mathbf{X})$, one can square ψ as to recover $p(\mathbf{x}) \propto \psi(\mathbf{x})^2$, yielding a model called *real* Born machine (BM) (Glasser et al. 2019). These models are popular in quantum physics and computing where they, however, operate over *complex* tensors. In our theoretical framework, it will be clear that complex parameters *do* bring an expressiveness advantage and *why* (Section 5). Before that, however, note that Theorem 0 does not say that squared PCs can efficiently represent *all* non-negative functions that monotonic PCs can compactly encode. In fact, our first theoretical contribution is showing that they cannot: we prove an exponential separation "in the other direction" next.

3 A Limitation of Squared Circuits

We show that there is a class of non-negative functions that can be efficiently represented by structured-decomposable monotonic circuits, whose class we denote as $+_{\mathsf{sd}}$, but for which any squared PC c^2 has exponential size. As we discussed before in Section 2, we use the property that c^2 is the output of the product algorithm $\mathsf{Multiply}$, and therefore is bounded by the size of the possibly negative circuit c: by constructing an exponential lower bound of the size of c, we therefore obtain a lower bound on c^2. While there could exist an alternative algorithm to directly construct a polynomial size c^2 that encodes our separating function, we conjecture this is unlikely. The next theorem formalizes our finding.

Theorem 1. There is a class of non-negative functions $\mathcal{F}$ over $d = k(k+1)$ variables that can be encoded by a PC in $+_{\mathsf{sd}}$ having size $\mathcal{O}(d)$. However, the smallest $c^2 \in \pm^2_{\mathbb{R}}$ computing any $F \in \mathcal{F}$ requires $|c|$ to be at least $2^{\Omega(\sqrt{d})}$.

Appendix B.2 details our proof: we leverage a lower bound of the square root rank of non-negative matrices (Fawzi et al. 2014) and we build the separating function family $\mathcal{F}$ starting from the *sum function* (SUM) defined as

$$\mathsf{SUM}(\mathbf{X}) = \sum\nolimits_{i=1}^{k} X_i \left(\sum\nolimits_{j=1}^{k} 2^{j-1} X_{i,j}\right), \tag{3}$$

where $\mathbf{X} = \{X_i \mid i \in [k]\} \cup \{X_{i,j} \mid (i,j) \in [k] \times [k]\}$ are Boolean variables. Our construction starts from the observation made by Glasser et al. (2019) that non-negative MPSs (Eq. (2)) encoding a non-negative factorization require an exponentially smaller rank r than real BMs to factorize a variant of our SUM function. Our result is stronger, as it holds for all squared structured-decomposable circuits, which implies for all possible tree-shaped region graphs. In fact, MPSs and BMs can only represent PCs with a certain region graph, as all product units in them decompose their scope by conditioning on one variable at a time (see Proposition 3 in Loconte et al. (2024a)). Our finding, therefore, *generalizes to other cases of squared tensor networks* over the reals, such as tree-shaped BMs (Shi, Duan, and Vidal 2006; Cheng et al. 2019) and many more circuits with tree region graphs (Mari, Vessio, and Vergari 2023).

With this in mind, Theorem 0 and Theorem 1 now tell us that PCs in $+_{\mathsf{sd}}$ and $\pm^2_{\mathbb{R}}$ exhibit different limitations on which functions they can encode. For this reason, we say that the circuit classes $+_{\mathsf{sd}}$ and $\pm^2_{\mathbb{R}}$ are *incomparable in terms of expressive efficiency* (de Colnet and Mengel 2021).

How can we surpass these limitations? A closer look at the SUM function in Eq. (3) motivates a circuit class that can be more expressive than $+_{\mathsf{sd}}$ and $\pm^2_{\mathbb{R}}$. We can rewrite it as $\mathsf{SUM}(\mathbf{X}) = \sum_{i=1}^{k}\sum_{j=1}^{k}(2^{(j-1)/2}X_iX_{i,j})^2$, exploiting the fact that Boolean variables are idempotent under powers. This new form, although semantically equivalent, highlights that one can simply encode SUM as a monotonic structured-decomposable circuit with a single sum unit over k^2 *squared* product units. Therefore, it suggests that even a sum of a few squared PCs together allows to efficiently encode more expressive functions that cannot be compactly captured by a *single* squared PC. To be more expressive than $+_{\mathsf{sd}}$, we need to relax monotonicity, instead. In the next section, we formalize this intuition by proposing a class of circuits that can be more expressive than those introduced so far, and unifies several other families of tractable models.

4 Sum of Compatible Squares Circuits

Our more expressive class of circuits will take the form of a *sum of squares* (SOS), a well-known concept in algebraic geometry where it is used to characterize some non-negative polynomials (Benoist 2017). As such, we derive a special SOS polynomial family that supports tractable inference, as circuits can also be understood as compact representations of polynomials whose indeterminates are the circuit input functions (Martens and Medabalimi 2014). As we are interested in precisely tracing the boundaries of expressive efficiency for circuits, we will require that our SOS forms always have polynomial size. This is in contrast with universality results for SOS and non-negative polynomials, where the interest is to prove that a non-negative polynomial can or cannot be written as a SOS, regardless of the model size. E.g., see the Hilbert's 17th problem and Marshall (2008) for a review. To preserve tractable inference, in our class of sum of squares circuits, we do not only require each circuit to be structured-decomposable as to efficiently square it, but also compatible with all the others.

Definition 4. A *sum of compatible squares* (SOCS) PC c over variables $\mathbf{X}$ is a PC encoding $c(\mathbf{x}) = \sum_{i=1}^{r} c_i^2(\mathbf{x})$ where, for all $i,j \in [r]$, $c_i^2, c_j^2 \in \pm^2_{\mathbb{R}}$ are compatible.

We denote this class of SOCS circuits as Σ^2_{cmp}. Combining many (squared) PCs in a positive sum is equivalent to building a finite mixture (McLachlan, Lee, and Rathnayake 2019) having (squared) PCs as components. Although this looks like a simple way to increase their expressiveness, we now show that that the advantage is in fact exponential.

Theorem 2. There is a class of non-negative functions $\mathcal{F}$ over d variables that can be represented by a PC in Σ^2_{cmp} of size $\mathcal{O}(d^3)$. However, (i) the smallest PC in $+_{\mathsf{sd}}$ computing any $F \in \mathcal{F}$ has at least size $2^{\Omega(\sqrt{d})}$, and (ii) the smallest $c^2 \in \pm^2_{\mathbb{R}}$ computing F obtained by squaring a structured-decomposable circuit c, requires $|c|$ to be at least $2^{\Omega(\sqrt{d})}$.

In fact, we prove *two* alternative exponential separations between Σ^2_{cmp} and *both* $+_{\mathsf{sd}}$ and $\pm^2_{\mathbb{R}}$ circuit classes, represented in Eqs. (4) and (5). Each combines a monotonic PC from $+_{\mathsf{sd}}$ and a squared one from $\pm^2_{\mathbb{R}}$ and thus provides a different insight on how to build more expressive circuits. Our proof of Theorem 2 uses our first separating function family $\mathcal{F}$, built as a *sum* of UDISJ (Eq. (1)) and SUM (Eq. (3)) embedded on a graph, and defined as $\mathsf{UPS}(\mathbf{X})$

$$= Z_1\bigl(1-\sum_{uv\in E} X_uX_v\bigr)^2 + Z_2\sum_{v\in V} X_v \sum\nolimits_{j=1}^{|V|} 2^{j-1}X_{v,j} \quad (4)$$

where $\mathbf{X} = \mathbf{X}' \cup \mathbf{X}'' \cup \{Z_1, Z_2\}$ are Boolean variables, with $\mathbf{X}' = \{X_v \mid v \in V\}$, $\mathbf{X}'' = \bigcup_{v\in V}\{X_{v,j} \mid j \in [|V|]\}$, and Z_1, Z_2 are auxiliary variables. By properly setting $Z_1 = 1, Z_2 = 0$ (resp. $Z_1 = 0, Z_2 = 1$) we retrieve a PC in $\pm^2_{\mathbb{R}}$ (resp. $+_{\mathsf{sd}}$). See Appendix B.3 for the proof details.

Theorem B.3 details our alternative exponential separation, where we let $\mathcal{F}$ be the *product* of UDISJ *times* a quadratic form that can be easily represented as a circuit in $+_{\mathsf{sd}}$. We name it $\mathsf{UTQ}(\mathbf{X})$ and define it on a graph as

$$\bigl(1-\sum\nolimits_{uv\in E} X_uX_v\bigr)^2\bigl(1+\sum\nolimits_{uv\in E} X_uX_v\bigr) \quad (5)$$

where $\mathbf{X} = \{X_v \mid v \in V\}$ are Boolean variables. A first advantage of this alternative construction over Theorem 2 is that Theorem B.3 provides the *strongly* exponential lower bound $2^{\Omega(d)}$ instead of the *weakly* exponential $2^{\Omega(\sqrt{d})}$ (Impagliazzo, Paturi, and Zane 1998). Furthermore, multiplying circuits from $+_{\mathsf{sd}}$ and $\pm^2_{\mathbb{R}}$ provides a perspective to better understand other existing tractable representations (Proposition 1) and build PCs that can compactly encode a sum of an exponential number of squares. To this end, we generalize this construction to a product of circuits in $+_{\mathsf{sd}}$ and Σ^2_{cmp}.

Definition 5. A *product of monotonic by SOCS* (μSOCS) PC c over variables $\mathbf{X}$ is a PC encoding $c(\mathbf{x}) = c_1(\mathbf{x})\cdot c_2(\mathbf{x})$, where $c_1 \in +_{\mathsf{sd}}$ and $c_2 \in \Sigma^2_{\mathsf{cmp}}$ are compatible.

μSOCS PCs can efficiently compute any function encoded by PCs in $+_{\mathsf{sd}}$ (resp. Σ^2_{cmp}), by taking as a SOCS PC (resp. structured monotonic PC) a circuit computing the constant 1. As a deep PC may compactly encode a polynomial with exponentially many terms (Martens and Medabalimi 2014) (which appears when we "unroll" such a PC (Proposition 3)), our μSOCS compactly encode a SOCS PC with an exponential number of squares. As such, we can gracefully scale SOCS to high-dimensional data, even multiplying a monotonic circuit by a single square (Section 7).

5 SOCS Circuits Unify Many Model Classes

We now extend the theoretical results presented so far to several other tractable model classes that can be reduced to SOCS. We start by models with complex parameters.

Complex parameters have been extensively used in ML. First, for their semantics as modeling waves, as in signal processing and physics (Hirose and Yoshida 2012; Tygert et al. 2015), E.g., we mentioned in Section 2 that MPSs and BMs are generally defined as encoding a factorization over the complexes. Thus, a BM in quantum physics models the *modulus squared* $|\psi(\mathbf{x})|^2 = \psi(\mathbf{x})^\dagger\psi(\mathbf{x})$, where $(\cdot)^\dagger$ denotes the complex conjugate operation of a complex function ψ, here factorized as a *complex* MPS. The tensors $\mathbf{A}_1, \ldots, \mathbf{A}_d$ shown in Eq. (2) are generalized to have complex-valued entries (Orús 2013). Complex BMs have been used for distribution estimation (Han et al. 2018; Cheng et al. 2019). Secondly, there is evidence supporting using complex parameters to stabilize learning (Arjovsky, Shah, and Bengio 2015) and boost performances of ML models, PCs included (Trouillon et al. 2016; Sun et al. 2019; Loconte et al. 2023). Within PCs, one can back up this evidence with a precise theoretical characterization and answer: *is there any expressiveness advantage in modeling PCs in the complex field?*

Complex squared PCs. We extend PCs in $\pm^2_{\mathbb{R}}$ and formally define a complex squared PC as the one computing $c^2(\mathbf{x}) = c(\mathbf{x})^\dagger c(\mathbf{x})$, i.e., via the modulus square of a structured-decomposable circuit c whose parameters and input functions are complex. Computing c^2 can be done efficiently in $\mathcal{O}(|c|^2)$ via $\mathsf{Multiply}(c^\dagger, c)$, where the complex conjugate $c^\dagger$ of c preserves the structural properties we need: smoothness and structured-decomposability as shown in Yu, Trapp, and Kersting (2023). We denote with $\pm^2_{\mathbb{C}}$ the class of complex squared PCs computing c^2. In Appendix B.5 we show that, similarly to the reduction of *real* BMs to squared PCs (Loconte et al. 2024a), *complex* BMs can be reduced to complex squared PCs, thus are at least as expressive as their real counterpart. Crucially, we prove that any complex squared PC can be efficiently reduced to a PC in Σ^2_{cmp} with real parameters only. This settles the question whether complex parameters bring an expressive advantage over reals through the lens of our Theorem 2. We formalize this next.

Corollary 1. Let $c^2 \in \pm^2_{\mathbb{C}}$ be a PC over variables $\mathbf{X}$, where c is a structured-decomposable circuit computing a complex function. Then, we can efficiently represent c^2 as the sum of two compatible PCs in $\pm^2_{\mathbb{R}}$, thus as a PC in Σ^2_{cmp}.

We generalize this result to squared PCs computing the modulus square of *hypercomplex numbers*, e.g., quaternions and generalizations (Shenitzer, Kantor, and Solodovnikov 1989). These have not only been studied in physics (Baez 2001), but also in ML (Saoud and Al-Marzouqi 2020; Yu et al. 2022). All these squared PCs are also SOCS.

Theorem 3. Let $\mathbb{A}_\omega$ be an algebra of hypercomplex numbers of dimension 2^ω (e.g., $\mathbb{A}_0 = \mathbb{R}$, $\mathbb{A}_1 = \mathbb{C}$, $\ldots$). Given a structured-decomposable circuit c over $\mathbf{X}$ computing $c(\mathbf{x}) \in \mathbb{A}_\omega$, we can efficiently represent a PC computing $c^2(\mathbf{x}) = c(\mathbf{x})^\dagger c(\mathbf{x})$ as a PC in Σ^2_{cmp} having size $\mathcal{O}(2^\omega|c|^2)$.

Appendix B.6 details our proof construction.

Squared neural families (SNEFYs) (Tsuchida, Ong, and Sejdinovic 2023, 2024) are recent estimators generalizing exponential families that model a distribution $p(\mathbf{X})$ as

$$\mathsf{SNEFY}_{t,\sigma,\mu}(\mathbf{x}) = Z^{-1}\mu(\mathbf{x})\, ||\mathsf{NN}_\sigma(t(\mathbf{x}))||_2^2$$

$t\colon \mathsf{dom}(\mathbf{X}) \to \mathbb{R}^J$ the sufficient statistics, NN_σ is a one-hidden-layer neural network with element-wise activation function σ and Z the partition function. Depending on the choice of t, σ and μ, $\mathsf{SNEFY}_{t,\sigma,\mu}$ allows tractable marginalization (Tsuchida, Ong, and Sejdinovic 2023). Next, we show that many common parameterizations of these tractable SNEFYs fall within Σ^2_{cmp}.

Proposition 1. Let $\mathsf{SNEFY}_{t,\sigma,\mu}$ be a distribution over d variables $\mathbf{X}$ with $\sigma \in \{\exp, \cos\}$, $\mu(\mathbf{x}) = \mu_1(x_1)\cdots\mu_d(x_d)$, $t(\mathbf{x}) = [t_1(x_1), \ldots, t_d(x_d)]^\top$. Then, $\mathsf{SNEFY}_{t,\sigma,\mu}$ can be encoded by a PC in Σ^2_{cmp} of size $\mathcal{O}(\mathrm{poly}(d, K))$, where K denotes the number of neural units in NN_σ.

We prove it in Appendix B.7. Note that the product of a factorized base measure $\mu(\mathbf{x}) = \mu_1(x_1)\cdots\mu_d(x_d)$ by the squared norm of a neural network (as in Proposition 1) can be written as the product between a *fully factorized monotonic PC* computing $\mu(\mathbf{x})$ and a SOCS PC. This is a special case of our μSOCS construction (Definition 5).

In their finite-dimensional form, **positive semi-definite (PSD) kernel models** (Marteau-Ferey, Bach, and Rudi 2020) are non-parametric estimators modeling $p(\mathbf{x}) \propto \boldsymbol{\kappa}(\mathbf{x})^\top \mathbf{A}\boldsymbol{\kappa}(\mathbf{x})$, where $\mathbf{A} \in \mathbb{R}^{r\times r}$ is a PSD matrix, and $\boldsymbol{\kappa}(\mathbf{x}) = [\kappa(\mathbf{x}, \mathbf{x}^{(1)}), \ldots, \kappa(\mathbf{x}, \mathbf{x}^{(r)})]^\top \in \mathbb{R}^r$ is a kernel defined over data points $\{\mathbf{x}^{(i)}\}_{i=1}^r$. For some choices of κ (e.g., an RBF kernel as in Rudi and Ciliberto (2021)), PSD models support tractable marginalization. Since PSD models are *shallow*, Sladek, Trapp, and Solin (2023) have recently proposed to combine them with *deep* monotonic PCs. They do so by parameterizing a sum unit n in a circuit to compute $c_n(\mathbf{x}) = c(\mathbf{x})^\top \mathbf{A} c(\mathbf{x})$, where $\mathbf{A}$ is a PSD matrix and c is a multi-output circuit computing an r-dimensional vector $c(\mathbf{x}) = [c_1(\mathbf{x}), \ldots, c_r(\mathbf{x})]^\top$, with $\{c_i\}_{i=1}^r$ being a set of non-monotonic circuits that are compatible with each other. From now on, we denote with psd the class of PCs computing $c(\mathbf{x})^\top \mathbf{A} c(\mathbf{x})$. Next, we show that PCs in psd *are equivalently expressive efficient* as PCs in Σ^2_{cmp}, as one can reduce any PC in the foremost class into the latter and vice versa.

Proposition 2. Any PC in psd can be reduced in polynomial time to a PC in Σ^2_{cmp}. The converse result also holds.

We prove it in Appendix B.8 by generalizing the reduction from *shallow* PSD kernel models shown by Loconte et al. (2024a). Sladek, Trapp, and Solin (2023) propose an alternative construction of deep PSD models where PSD sum units are at the inputs of a monotonic PC, instead of its output. In this way, the number of squares they are summing over is exponential in the circuit depth, akin to our μSOCSs.

Inception PCs have been concurrently introduced to our SOCSs as non-monotonic PCs overcoming the limitations of squared PCs in Wang and Van den Broeck (2024), where an alternative version of our proof that structured-decomposable monotonic circuits can be exponentially more expressive than a single squared circuit with real parameters (i.e., our Theorem 1) was provided. These results were submitted to a conference around the same time as ours. An Inception PC encodes a distribution p over variables $\mathbf{X}$ as

$$p(\mathbf{x}) \propto \sum\nolimits_{\mathbf{u}\in\mathsf{dom}(\mathbf{U})} \Big|\sum\nolimits_{\mathbf{w}\in\mathsf{dom}(\mathbf{W})} c_{\mathsf{aug}}(\mathbf{x}, \mathbf{u}, \mathbf{w})\Big|^2, \quad (6)$$

where c_{aug} is a structured circuit computing a complex function over a set of augmented variables $\mathbf{X} \cup \mathbf{U} \cup \mathbf{W}$. Eq. (6) is already written in a SOCS form. Similarly to PSD circuits with PSD inputs and our μSOCSs, summing $\mathbf{U}$ outside the square can compactly represent an exponential number of squares through parameter sharing. Differently from our μSOCS, however, Inception PCs suffer a quadratic blow up even to compute the unnormalized log-likelihood as they cannot push the square outside the logarithm (see Section 7).

6 Are All Structured PCs SOCS Circuits?

Given the increased expressiveness of SOCS PCs as discussed so far, a natural question now is whether they can *efficiently encode any distribution computed by other tractable PC classes*. Clearly, we need to restrict our attention to structured-decomposable circuits, as this structural property is required for efficient squaring. This rules out from our analysis circuits that are just decomposable or those that encode multilinear polynomials but are not decomposable. (Agarwal and Bläser 2024; Broadrick, Zhang, and Van den Broeck 2024). We start by focusing on PCs in $+_{\mathsf{sd}}$, and show an upper bound on the size of SOCS PCs computing them.

Proposition 3. Every function over d Booleans computed by a PC in $+_{\mathsf{sd}}$ can be encoded by one in Σ^2_{cmp} of size $\mathcal{O}(2^d)$.

Our proof in Appendix B.9 rewrites the circuit as a shallow polynomial and exploits idempotency of Boolean variables w.r.t. powers. This is not surprising, as every non-negative Boolean polynomial is known to be a SOS polynomial (Barak and Steurer 2016). However, it is unknown if there exists a sub-exponential size upper bound for a SOCS PC computing PCs in $+_{\mathsf{sd}}$, leading to our first open question.

Open Question 1. Can any function computed by a polysize PC in $+_{\mathsf{sd}}$ be also computed by a polysize PC in Σ^2_{cmp}?

If we extend our analysis to PCs with non-Boolean inputs, we connect with another classical result in algebraic geometry showing there are many non-negative polynomials that *cannot be written as SOS polynomials* with real coefficients (Blekherman 2003). E.g., Motzkin polynomial (Motzkin 1967) is defined as the bivariate polynomial

$$F_{\mathcal{M}}(X_1, X_2) = 1 + X_1^4 X_2^2 + X_1^2 X_2^4 - 3X_1^2 X_2^2 \qquad (7)$$

that while is non-negative over its domain $\mathbb{R}^2$, is also known *not* to be a SOS (Marshall 2008). We generalize Motzkin polynomial to create a class of polynomials on an arbitrary number of variables that can be computed by PCs in $\pm_{\mathsf{sd}}$ when equipped with polynomials as input functions but cannot be computed by SOCS PCs with the same inputs.

Theorem 4. There exists a class of non-negative functions $\mathcal{F}$ over d variables, that cannot be computed by SOCS PCs whose input units encode polynomials. However, for any $F \in \mathcal{F}$, there exists a PC in $\pm_{\mathsf{sd}}$ of size $\mathcal{O}(d^2)$ with polynomials as input units computing it.

Appendix B.10 details our proof. Note that the limitation shown in Theorem 4 remains even if we relax the compatibility assumption for our SOCS (Definition 4) and obtain a new class of *sum of structured-decomposable squares circuits*. However, if we do not make any assumption on the functions computed by the input units of SOCS PCs, it remains open to formally show whether any function computed by a structured-decomposable PC can be efficiently computed by a PC in Σ^2_{cmp}, as formalized below.

Open Question 2. Can any function computed by a polysize PC in $\pm_{\mathsf{sd}}$ be also computed by a polysize PC in Σ^2_{cmp}?

Answering affirmatively to Open Question 2 would also solve Open Question 1, since $+_{\mathsf{sd}} \subset \pm_{\mathsf{sd}}$. We visualize both open questions in our expressive efficiency hierarchy in Fig. 1. While we do not answer to Open Question 2, the following theorem shows that a single subtraction between SOCS PCs is however sufficient to encode any function computed by a structured-decomposable circuit.

Theorem 5. Let c be a structured circuit over $\mathbf{X}$, where $\mathsf{dom}(\mathbf{X})$ is finite. The (possibly negative) function computed by c can be encoded in worst-case time and space $\mathcal{O}(|c|^3)$ as the difference $c(\mathbf{x}) = c_1(\mathbf{x}) - c_2(\mathbf{x})$ with $c_1, c_2 \in \Sigma^2_{\mathsf{cmp}}$.

We prove it in Appendix B.11. We denote as $\Delta\Sigma^2_{\mathsf{cmp}}$ the class of PCs obtained by subtracting two PCs in Σ^2_{cmp}. Since $\Delta\Sigma^2_{\mathsf{cmp}} \subset \pm_{\mathsf{sd}}$, Theorem 5 implies the expressive efficiency equivalence between classes $\Delta\Sigma^2_{\mathsf{cmp}}$ and $\pm_{\mathsf{sd}}$, in the case of finite variables domain. Our result is similar to a classical result in Valiant (1979) but we consider circuits with different properties: they show a (non-decomposable) non-monotonic circuit is computable as the difference of monotonic ones, we focus on structured circuits rewritten as SOCS PCs.

7 Empirical Evaluation

We evaluate structured monotonic ($+_{\mathsf{sd}}$), squared PCs ($\pm^2_{\mathbb{R}}$, $\pm^2_{\mathbb{C}}$), their sums and μSOCS as the product of a monotonic and a SOCS PC ($+_{\mathsf{sd}} \cdot \Sigma^2_{\mathsf{cmp}}$, see Definition 5) on distribution estimation tasks using both continuous and discrete real-world data. We answer to the following questions: **(A)** how does a monotonic PC perform with respect to a *single* squared PC with the same model size? **(B)** how does increasing the number of squares in a SOCS PC influence its expressiveness? **(C)** how do SOCS PCs perform w.r.t. monotonic PCs as we scale them on high-dimensional image data?

Experimental setting. Given a training set $\mathcal{D} = \{\mathbf{x}^{(i)}\}_{i=1}^N$ on variables $\mathbf{X}$, we are interested in estimating $p(\mathbf{X})$ from $\mathcal{D}$ by minimizing the parameters negative log-likelihood on a batch $\mathcal{B} \subset \mathcal{D}$, i.e., $\mathcal{L} := |\mathcal{B}| \log Z - \sum_{\mathbf{x}\in\mathcal{B}} \log c(\mathbf{x})$, via gradient descent. For squared PCs in $\pm^2_{\mathbb{R}}$ and $\pm^2_{\mathbb{C}}$, we can compute Z just once per batch as done in Loconte et al. (2024a), making training particularly efficient. We can use the same trick for sums of $\pm^2_{\mathbb{R}}$ or $\pm^2_{\mathbb{C}}$ PCs by rewriting $\mathcal{L}$ as $\mathcal{L} := |\mathcal{B}| \log Z - \sum_{\mathbf{x}\in\mathcal{B}} \log \sum_{i=1}^r \exp(2 \log |c_i(\mathbf{x})|)$, thus requiring materializing the squared PCs $c_1^2, \ldots, c_r^2$ only to compute Z. It also applies to μSOCS as the circuit product decomposes into a sum of logs (see Appendix C.3). PCs are compared based on the average log-likelihood on unseen data points. Next, we briefly discuss how we build PCs, and refer the reader to Appendix C for more details.

Building tensorized SOCS PCs. Following Mari, Vessio, and Vergari (2023), we build tensorized PCs by parameterizing region graphs (Section 2) with sum and product lay-

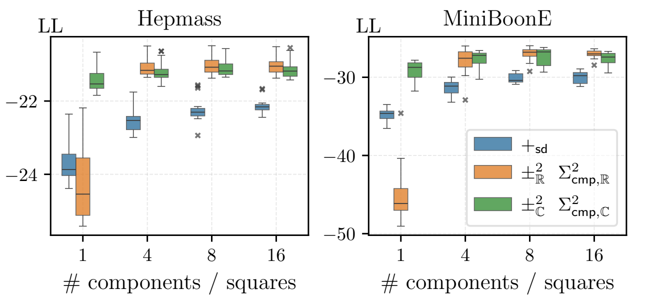

Figure 2: Monotonic PCs ($+_{\mathsf{sd}}$) can perform better than a *single* real squared PC ($\pm^2_{\mathbb{R}}$) on average, but worse than a *single* complex squared PC ($\pm^2_{\mathbb{C}}$), and worse than SOCS PCs ($\Sigma^2_{\mathsf{cmp},\mathbb{R}}$ and $\Sigma^2_{\mathsf{cmp},\mathbb{C}}$) with an increasing number of squares. For $+_{\mathsf{sd}}$ we take mixtures of monotonic PC as components. We show box-plots of average test log-likelihoods on multiple runs. All PCs have approximately the same number of parameters (see main text). Details in Appendix C.2.

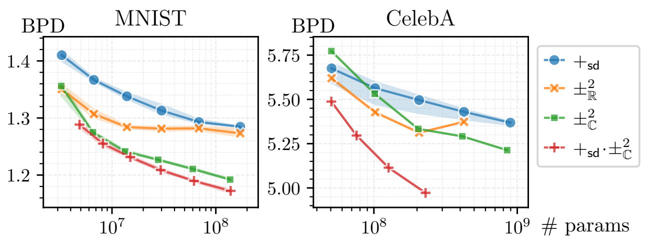

Figure 3: Complex squared PCs are more accurate estimators on image data. We show test BPD (lower is better) w.r.t. the number of learnable parameters of structured monotonic PCs ($+_{\mathsf{sd}}$), squared PCs with real ($\pm^2_{\mathbb{R}}$) and complex ($\pm^2_{\mathbb{C}}$) parameters (which are counted twice), and the product of a monotonic PC by a complex squared PC ($+_{\mathsf{sd}} \cdot \pm^2_{\mathbb{C}}$, see Definition 5). We report the area between min and max BPDs obtained from 5 independent runs with different seeds.

ers forming a CP decomposition and vectorized input layers (Loconte et al. 2024b). This construction allows us to govern model size by selecting how many units are placed in layers. We experiment with two types of region graphs depending on the data: for tabular data, we follow Loconte et al. (2024a) and use random binary trees; for images, we use *quad trees*, that recursively split images into patches (see Appendix C.2). Implementing our complex PCs can be done by upcasting the tensor types and implementing a variant of the *log-sum-exp trick* on complex numbers (Appendix C.4).

(A) We estimate the distribution of four continuous UCI data sets: Power, Gas, Hepmass, MiniBooNE, using the same preprocessing by Papamakarios, Pavlakou, and Murray (2017) (Table C.1). We compare structured monotonic PCs, and squared PCs with real or complex parameters. All PCs use Gaussian likelihoods as input units, and we ensure they have the same number of trainable parameters up to a $\pm 6\%$ difference. Figs. 2 and C.1 show that a monotonic PC achieves higher test log-likelihoods than a single squared PC with *real* parameters on all data sets but Gas, thus agreeing with our theory stating an expressiveness limitation of a single *real* squared PC (Theorem 1). Instead, complex squared PCs consistently perform better than both a single real squared PC and a monotonic PC, as they are SOCS (Corollary 1). Fig. C.2 shows the training log-likelihoods. Finally, Fig. C.3 shows that learning a single real squared PC is typically challenging due to an unstable loss, which is instead not observed for complex squared PCs.

(B) We now gradually increase the number of squares from 4 to 16 in SOCS PCs, while still keeping the number of parameters approximately the same. As a baseline, we build mixtures of 4-16 structured monotonic PCs. Our comparisons are performed in the same setting and data sets of **(A)**. Fig. 2 shows that not only SOCS PCs outperform monotonic PCs, but also that summing more than 4 squared PCs does not bring a significant benefit on these datasets. See Fig. C.1 for results on Power and Gas. Moreover, complex SOCS PCs perform similarly to real SOCS PCs. This is expected as complex SOCS PCs belong to the class Σ^2_{cmp} (Corollary 1).

(A, C) We estimate the probability distribution of MNIST, FashionMNIST and CelebA images (Table C.2), and compare a single structured monotonic PC, a single squared PC with real and complex parameters and a μSOCS PC created by multiplying a monotonic PC and squared complex one ($+_{\mathsf{sd}} \cdot \pm^2_{\mathbb{C}}$). Appendix C.2 and C.3 show our model constructions. For monotonic PCs, we use Categorical distributions as inputs, while for real (resp. complex) squared PCs we use real (resp. complex) vector embeddings. Fig. 3 reports the normalized negated log-likelihood as *bits-per-dimension* (BPD). There, complex squared PCs better estimate the image distribution than monotonic PCs and real squared PCs. Our μSOCS fares better than a single complex squared PC and with fewer parameters. The performance of real squared PCs, instead, plateaus and becomes comparable to monotonic PCs. Fig. C.4 shows similar trends on FashionMNIST and the training BPDs. While both real and complex squared PCs tend to overfit, complex squared PCs and μSOCS generalize better. Appendix C.6 compares complex squared PCs and μSOCS in terms of training time and space.

8 Conclusion

We unified and separated several recent tractable models built around the circuit squaring operation. Our theoretical characterization justifies many claims supported only by empirical evidence, e.g., the use of complex parameters. Our SOCS PCs deliver not only scalable and accurate models but also enable further connections with the literature on SOS polynomials. Our SOCS expressiveness results can be translated to the literature of non-linear matrix decompositions (Lefebvre, Vandaele, and Gillis 2024; Awari et al. 2024). As future direction, we plan to retrieve a rigorous latent variable interpretation for SOCS (Peharz et al. 2016), thus unlocking parameter and structure learning schemes (Gens and Domingos 2013; Di Mauro et al. 2017). Moreover, we plan to extend SOCS to continuous latent variables (Gala et al. 2024a,b) and use neural networks to parameterize them (Shao et al. 2020) as to further increase their expressiveness.

Acknowledgments

We would like to acknowledge Nicolas Gillis for insightful discussions on the square root rank of non-negative matrices, as well as on the properties of the uniqueness disjointness function related to the non-negative matrix rank. We acknowledge Raul Garcia-Patron Sanchez for meaningful discussions about tensor networks and quantum circuits. We thank Rickey Liang for suggesting performing computations using the complex logarithm and complex exponential definitions to ensure numerical stability in our experiments. AV was supported by the "UNREAL: Unified Reasoning Layer for Trustworthy ML" project (EP/Y023838/1) selected by the ERC and funded by UKRI EPSRC.

Contributions

LL and AV conceived the initial idea of the paper. LL is responsible for all theoretical contributions with the following exceptions. SM provided Theorem 2 as an alternative lower bound to Theorem B.3 and simplified the proofs of Theorem 1 and Theorem 4 that were originally proven by LL. LL developed the necessary code, run all the experiments, plotted the figures and wrote the paper with the help of AV. AV supervised all the phases of the project and provided feedback. All authors revised the manuscript critically.

References

Agarwal, S.; and Bläser, M. 2024. Probabilistic Generating Circuits – Demystified. In *ICML*.

Ahmed, K.; Teso, S.; Chang, K.-W.; Van den Broeck, G.; and Vergari, A. 2022. Semantic probabilistic layers for neuro-symbolic learning. In *NeurIPS*.

Arjovsky, M.; Shah, A.; and Bengio, Y. 2015. Unitary Evolution Recurrent Neural Networks. In *ICML*.

Awari, A.; Nguyen, H.; Wertz, S.; Vandaele, A.; and Gillis, N. 2024. Coordinate Descent Algorithm for Nonlinear Matrix Decomposition with the ReLU function. In *EUSIPCO*.

Baez, J. 2001. The Octonions. *Bulletin of the American Mathematical Society*.

Barak, B.; and Steurer, D. 2016. Proofs, beliefs, and algorithms through the lens of sum-of-squares. https://www.sumofsquares.org/public/index.html.

Benoist, O. 2017. Writing Positive Polynomials as Sums of (Few) Squares. *European Mathematical Society Magazine*.

Blekherman, G. 2003. There are significantly more nonegative polynomials than sums of squares. *Israel Journal of Mathematics*.

Broadrick, O.; Zhang, H.; and Van den Broeck, G. 2024. Polynomial Semantics of Tractable Probabilistic Circuits. In *UAI*.

Cheng, S.; Wang, L.; Xiang, T.; and Zhang, P. 2019. Tree tensor networks for generative modeling. *Physical Review*.

Choi, Y.; Vergari, A.; and Van den Broeck, G. 2020. Probabilistic Circuits: A Unifying Framework for Tractable Probabilistic Modeling. Technical report, University of California, Los Angeles (UCLA).

Darwiche, A.; and Marquis, P. 2002. A knowledge compilation map. *Journal of Artificial Intelligence Research (JAIR)*.

de Colnet, A.; and Mengel, S. 2021. A Compilation of Succinctness Results for Arithmetic Circuits. In *18th International Conference on Principles of Knowledge Representation and Reasoning (KR)*.

De Wolf, R. 2003. Nondeterministic quantum query and communication complexities. *SIAM Journal on Computing*.

Dechter, R.; and Mateescu, R. 2007. AND/OR search spaces for graphical models. *Artificial intelligence*.

Dennis, A. W. 2016. *Algorithms for Learning the Structure of Monotone and Nonmonotone Sum-Product Networks*. Ph.D. thesis, Brigham Young University.

Di Mauro, N.; Vergari, A.; Basile, T. M. A.; and Esposito, F. 2017. Fast and Accurate Density Estimation with Extremely Randomized Cutset Networks. In *ECML/PKDD*.

Fawzi, H.; Gouveia, J.; Parrilo, P. A.; Robinson, R. Z.; and Thomas, R. R. 2014. Positive semidefinite rank. *Mathematical Programming*.

Gala, G.; de Campos, C.; Peharz, R.; Vergari, A.; and Quaeghebeur, E. 2024a. Probabilistic Integral Circuits. In *AISTATS 2024*.

Gala, G.; de Campos, C.; Vergari, A.; and Quaeghebeur, E. 2024b. Scaling Continuous Latent Variable Models as Probabilistic Integral Circuits. *arXiv preprint arXiv:2406.06494*.

Gens, R.; and Domingos, P. M. 2013. Learning the Structure of Sum-Product Networks. In *ICML*.

Glasser, I.; Sweke, R.; Pancotti, N.; Eisert, J.; and Cirac, I. 2019. Expressive power of tensor-network factorizations for probabilistic modeling. In *NeurIPS*.

Han, Z.-Y.; Wang, J.; Fan, H.; Wang, L.; and Zhang, P. 2018. Unsupervised Generative Modeling Using Matrix Product States. *Physical Review X*.

Hirose, A.; and Yoshida, S. 2012. Generalization Characteristics of Complex-Valued Feedforward Neural Networks in Relation to Signal Coherence. *IEEE Transactions on Neural Networks and Learning Systems*.

Impagliazzo, R.; Paturi, R.; and Zane, F. 1998. Which problems have strongly exponential complexi.ty? *39th Annual Symposium on Foundations of Computer Science*.

Lefebvre, J.; Vandaele, A.; and Gillis, N. 2024. Component-wise Squared Factorization. In *IEEE International Workshop on Machine Learning for Signal Processing*.

Loconte, L.; Aleksanteri, M. S.; Mengel, S.; Trapp, M.; Solin, A.; Gillis, N.; and Vergari, A. 2024a. Subtractive Mixture Models via Squaring: Representation and Learning. In *ICLR*.

Loconte, L.; Di Mauro, N.; Peharz, R.; and Vergari, A. 2023. How to Turn Your Knowledge Graph Embeddings into Generative Models via Probabilistic Circuits. In *NeurIPS*.

Loconte, L.; Mari, A.; Gala, G.; Peharz, R.; de Campos, C.; Quaeghebeur, E.; Vessio, G.; and Vergari, A. 2024b. What is the Relationship between Tensor Factorizations and Circuits (and How Can We Exploit it)? *arXiv preprint arXiv:2409.07953*.

Marconato, E.; Bortolotti, S.; van Krieken, E.; Vergari, A.; Passerini, A.; and Teso, S. 2024a. BEARS Make Neuro-Symbolic Models Aware of their Reasoning Shortcuts. In *UAI*.

Marconato, E.; Teso, S.; Vergari, A.; and Passerini, A. 2024b. Not all neuro-symbolic concepts are created equal: Analysis and mitigation of reasoning shortcuts. *NeurIPS*.

Mari, A.; Vessio, G.; and Vergari, A. 2023. Unifying and Understanding Overparameterized Circuit Representations via Low-Rank Tensor Decompositions. In *6th Workshop on Tractable Probabilistic Modeling*.

Marshall, M. 2008. *Positive polynomials and sums of squares*. 146. American Mathematical Soc.

Marteau-Ferey, U.; Bach, F.; and Rudi, A. 2020. Non-parametric models for non-negative functions. In *NeurIPS*.

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

McLachlan, G. J.; Lee, S. X.; and Rathnayake, S. I. 2019. Finite mixture models. *Annual Review of Statistics and its Application*.

Motzkin, T. S. 1967. The arithmetic-geometric inequality. In *Proceedings Symposium Wright-Patterson Air Force Base, Ohio*. Academic Press, New York.

Orús, R. 2013. A Practical Introduction to Tensor Networks: Matrix Product States and Projected Entangled Pair States. *Annals of Physics*.

Oseledets, I. V. 2011. Tensor-Train Decomposition. *SIAM Journal on Scientific Computing*.

Papamakarios, G.; Pavlakou, T.; and Murray, I. 2017. Masked Autoregressive Flow for Density Estimation. In *NeurIPS*.

Peharz, R.; Gens, R.; Pernkopf, F.; and Domingos, P. 2016. On the latent variable interpretation in sum-product networks. *IEEE transactions on pattern analysis and machine intelligence*.

Pérez-García, D.; Verstraete, F.; Wolf, M. M.; and Cirac, J. I. 2007. Matrix Product State Representations. *Quantum Information and Computing*.

Pipatsrisawat, K.; and Darwiche, A. 2008. New Compilation Languages Based on Structured Decomposability. In *AAAI*.

Rudi, A.; and Ciliberto, C. 2021. PSD Representations for Effective Probability Models. In *NeurIPS*.

Saoud, L. S.; and Al-Marzouqi, H. 2020. Metacognitive Sedenion-Valued Neural Network and its Learning Algorithm. *IEEE Access*.

Shao, X.; Molina, A.; Vergari, A.; Stelzner, K.; Peharz, R.; Liebig, T.; and Kersting, K. 2020. Conditional sum-product networks: Imposing structure on deep probabilistic architectures. In *International Conference on Probabilistic Graphical Models*. PMLR.

Shenitzer, A.; Kantor, I.; and Solodovnikov, A. 1989. *Hypercomplex Numbers: An Elementary Introduction to Algebras*. Springer New York. ISBN 9780387969800.

Shi, Y.-Y.; Duan, L.-M.; and Vidal, G. 2006. Classical simulation of quantum many-body systems with a tree tensor network. *Physical Review*.

Shpilka, A.; and Yehudayoff, A. 2010. Arithmetic Circuits: A survey of recent results and open questions. *Founddations and Trends in Theoretical Computer Science*.

Sladek, A.; Trapp, M.; and Solin, A. 2023. Encoding Negative Dependencies in Probabilistic Circuits. In *6th Workshop on Tractable Probabilistic Modeling*.

Sun, Z.; Deng, Z.; Nie, J.; and Tang, J. 2019. RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space. In *ICLR*.

Trouillon, T.; Welbl, J.; Riedel, S.; Gaussier, E.; and Bouchard, G. 2016. Complex Embeddings for Simple Link Prediction. In *ICML*.

Tsuchida, R.; Ong, C. S.; and Sejdinovic, D. 2023. Squared Neural Families: A New Class of Tractable Density Models. In *NeurIPS*.

Tsuchida, R.; Ong, C. S.; and Sejdinovic, D. 2024. Exact, Fast and Expressive Poisson Point Processes via Squared Neural Families. In *Proceedings of the AAAI Conference on Artificial Intelligence*. AAAI Press.

Tygert, M.; Bruna, J.; Chintala, S.; LeCun, Y.; Piantino, S.; and Szlam, A. 2015. A Mathematical Motivation for Complex-Valued Convolutional Networks. *Neural Computation*.

Valiant, L. G. 1979. Negation can be exponentially powerful. In *11th ACM Symposium on Theory of Computing*.

Vergari, A.; Choi, Y.; Liu, A.; Teso, S.; and Van den Broeck, G. 2021. A Compositional Atlas of Tractable Circuit Operations for Probabilistic Inference. In *NeurIPS*.

Vergari, A.; Di Mauro, N.; and Van den Broeck, G. 2019. Tractable probabilistic models: Representations, algorithms, learning, and applications. *UAI*.

Wang, B.; and Van den Broeck, G. 2024. On the Relationship Between Monotone and Squared Probabilistic Circuits. arXiv:2408.00876.

Yu, M.; Bai, C.; Yu, J.; Zhao, M.; Xu, T.; Liu, H.; Li, X.; and Yu, R. 2022. Translation-Based Embeddings with Octonion for Knowledge Graph Completion. *Applied Sciences*.

Yu, Z.; Trapp, M.; and Kersting, K. 2023. Characteristic Circuits. In *NeurIPS*.