Continuous Mixtures of Tractable Probabilistic Models

Alvaro H.C. Correia[1,*], **Gennaro Gala**[1,*],
Erik Quaeghebeur[1], **Cassio de Campos**[1], **Robert Peharz**[1,2]

[1] Eindhoven University of Technology
[2] Graz University of Technology
{a.h.chaim.correia, g.gala, e.quaeghebeur, c.decampos, r.peharz}@tue.nl

Abstract

Probabilistic models based on continuous latent spaces, such as variational autoencoders, can be understood as uncountable mixture models where components depend continuously on the latent code. They have proven to be expressive tools for generative and probabilistic modelling, but are at odds with tractable probabilistic inference, that is, computing marginals and conditionals of the represented probability distribution. Meanwhile, tractable probabilistic models such as probabilistic circuits (PCs) can be understood as hierarchical discrete mixture models, and thus are capable of performing exact inference efficiently but often show subpar performance in comparison to continuous latent-space models. In this paper, we investigate a hybrid approach, namely continuous mixtures of tractable models with a small latent dimension. While these models are analytically intractable, they are well amenable to numerical integration schemes based on a finite set of integration points. With a large enough number of integration points the approximation becomes de-facto exact. Moreover, for a finite set of integration points, the integration method effectively compiles the continuous mixture into a standard PC. In experiments, we show that this simple scheme proves remarkably effective, as PCs learnt this way set new state of the art for tractable models on many standard density estimation benchmarks.

Introduction

Probabilistic modelling typically aims to capture the data-generating joint distribution, which can then be used to perform probabilistic inference to answer queries of interest. A recurring scheme in probabilistic modelling is the use of an *uncountable mixture model*, that is, the data generating distribution is approximated by

$$p(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})}\left[p(\mathbf{x} \mid \mathbf{z})\right] = \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z} \qquad (1)$$

where $p(\mathbf{z})$ is a mixing distribution (prior) over latent variables $\mathbf{Z}$, $p(\mathbf{x} \mid \mathbf{z})$ is a conditional distribution of $\mathbf{x}$ given $\mathbf{z}$ (mixture components), and $p(\mathbf{x})$ is the modelled density over variables $\mathbf{X}$, given by marginalising $\mathbf{Z}$ from the joint distribution defined by $p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})$.

Some successful recent examples of uncountable mixtures are variational autoencoders (VAEs) (Kingma and Welling 2014), generative adversarial networks (GANs) (Goodfellow et al. 2014), and normalising Flows (Rezende and Mohamed 2015). All three of these models use a simple prior $p(\mathbf{z})$, e.g. an isotropic Gaussian, and represent the mixture components with a neural network. In the case of VAEs, the mixture component is a proper density $p(\mathbf{x} \mid \mathbf{z})$ with respect to the Lebesgue measure, represented by the so-called decoder, while for GANs and Flows the mixture component is a point measure, i.e. a deterministic function $\mathbf{x} = f(\mathbf{z})$[1]. The use of continuous neural networks topologically relates the latent space and the observable space with each other, so that these models can be described as *continuous* mixture models. The use of continuous mixtures allows, to a certain extent, the interpretation of $\mathbf{Z}$ as a (latent) embedding of $\mathbf{X}$, but also seems to benefit generalisation, i.e. to faithfully approximate real-world distributions with limited training data.

However, while continuous mixture models have achieved impressive results in density estimation and generative modelling, their ability to support probabilistic inference remains limited. Notably, the key inference routines of *marginalisation* and *conditioning*, which together form a consistent reasoning process (Ghahramani 2015; Jaynes 2003), are largely intractable in these models, mainly due to the integral in (1) which forms a hard computational problem in general.

Meanwhile, the area of *tractable probabilistic modelling* aims at models which allow a wide range of exact and efficient inference routines. One of the most prominent frameworks to express tractable models are *probabilistic circuits* (PCs) (Vergari et al. 2020), which are computational graphs composed of (simple) tractable input distributions, factorisations (product nodes) and discrete mixture distributions (sum nodes). PCs describe many tractable models such as *Chow-Liu trees* (Chow and Liu 1968), *arithmetic circuits* (Darwiche 2003), *sum-product networks* (Poon and Domingos 2011), *cutset networks* (Rahman, Kothalkar, and Gogate 2014), *probabilistic sentential decision diagrams* (Kisa et al. 2014), and *generative forests* (Correia, Peharz, and de Campos 2020). The distribution represented by a PC depends both on its network structure $\mathcal{S}$ and its parameters ϕ, which

[*] These authors contributed equally.

[1] In Flows $\mathbf{z}$ is not truly latent since it relates to $\mathbf{x}$ via a bijection.

An extended version of this paper that includes the Appendix is available at arxiv.org/abs/2209.10584.

contains all weights of its sum nodes and parameters of its input distributions.

From a representational point of view, PCs can be interpreted as hierarchical, discrete mixture models (Peharz et al. 2016; Zhao, Poupart, and Gordon 2016), i.e. they can be generally written as

$$p(\mathbf{x}) = \sum_{\mathbf{z}'} p(\mathbf{x} \,|\, \mathbf{z}')\, p(\mathbf{z}') \tag{2}$$

where $\mathbf{Z}'$ is a *discrete* latent vector, but otherwise the form is similar to the continuous mixture in (1). The number of states of $\mathbf{Z}'$ and thus the number of represented mixture components $p(\mathbf{x} \,|\, \mathbf{z}')$ grows exponentially in the depth of the PC (Peharz 2015; Zhao, Poupart, and Gordon 2016). Moreover, recent vectorisation-based implementations (Peharz et al. 2020a) have enabled large PC architectures (>100M of parameters) at execution speeds comparable to standard neural networks. These endeavours evidently boosted the performance of tractable models, yet there is still a notable gap to intractable models like VAEs. One reason for this performance gap is likely the structural constraints in PCs, which are required to maintain tractability but are at odds with expressivity. On the other hand, a *huge* discrete mixture model of the form (2) should in principle be able to outperform a moderately sized uncountable mixture (1). Yet, on standard benchmarks, we do not see this result. For instance, a vanilla VAE with a few million parameters gets test log-likelihoods higher than -90 nats on Binary MNIST (Tomczak and Welling 2018), while an Einet with 84 million parameters (Peharz et al. 2020a) barely gets above -100 nats (or 0.184 bpd) as shown in Table 3. Thus, a complementary explanation is that discrete (and hierarchical) mixtures—like PCs—are hard to learn (or generalise poorly), while continuous mixtures—like VAEs—are easier to learn (or generalise well).

In this paper, we follow a hybrid approach and consider *continuous mixtures of tractable models.* In particular, we consider continuous mixtures of two very simple tractable models, namely *completely factorised distributions* and *tree-shaped graphical models*, which can both be easily expressed as PCs. Specifically, we consider models of the form

$$p(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})} \left[p(\mathbf{x} \,|\, \phi(\mathbf{z})) \right], \tag{3}$$

where $p(\mathbf{z})$ is an isotropic Gaussian prior and $p(\mathbf{x} \,|\, \phi(\mathbf{z}))$ is a PC with its parameters depending on $\mathbf{z}$ via some neural-network $\phi(\mathbf{z})$. That has certain parallels to previous works in *Conditional SPNs* (Shao et al. 2020) and *HyperSPNs* (Shih, Sadigh, and Ermon 2021). While continuous mixtures are analytically intractable, we can approximate the marginalisation of $\mathbf{z}$ arbitrarily well with numerical techniques, such as *Gaussian quadrature rules* and *(quasi) Monte Carlo*. The common principle of these methods is that they select a finite set of *integration points* $\mathbf{z}_1, \ldots, \mathbf{z}_N$ in either a deterministic or (partially) random manner and construct a corresponding weight function $w(\mathbf{z})$ such that (3) is approximated as

$$p(\mathbf{x}) = \mathbb{E}_{p(\mathbf{z})} \left[p(\mathbf{x} \,|\, \phi(\mathbf{z})) \right] \approx \sum_{i=1}^{N} w(\mathbf{z}_i)\, p(\mathbf{x} \,|\, \phi(\mathbf{z}_i)). \tag{4}$$

All integration methods we consider become exact for $N \to \infty$ and, under certain conditions on $\phi(\mathbf{z})$, one can derive guarantees for the approximation error for $N < \infty$. In particular, for (quasi) Monte Carlo integration it is straightforward to derive probabilistic error guarantees for the approximation quality by leveraging concentration bounds. Moreover, an empirical observation is that numerical integration works reasonably well for low dimensional spaces, but tends to deteriorate for larger dimensionality. Thus, in this paper we keep the dimensionality of $\mathbf{Z}$ relatively small (≤ 16), so that our continuous mixtures of tractable models remain 'morally tractable'.

Specifically, the integration weights $\{w(\mathbf{z}_i)\}_{i=1}^N$ typically sum to one[2], so that the approximation in (4) can be interpreted as a discrete mixture model. For fixed $\mathbf{z}_i$, each $p(\mathbf{x} \,|\, \phi(\mathbf{z}_i))$ is simply a PC with fixed parameters $\phi_i = \phi(\mathbf{z}_i)$, so that (4) is in fact a mixture of PCs, which in turn can be interpreted as a larger PC (Vergari et al. 2020). Thus, we can convert a learnt intractable model from (3) into a PC which facilitates exact inference, that is, 'performing exact inference in an accurately approximate model'.

To the best of our knowledge, this simple idea for constructing tractable models has not been explored before. Yet, it delivers astonishing results: on 16 out of 20 commonly used density estimation benchmarks, we set new state of the art (SOTA) among tractable models, outperforming even the most sophisticated PC structure and parameter learners. We also achieve competitive results on image datasets, where in comparison to other PCs, our models produced better samples and often attained better test log-likelihoods.

Background and Related Work

In this paper we are interested in *tractable* probabilistic models, i.e. models which allow for exact and efficient inference. *Probabilistic circuits* (PCs) are a prominent language for tractable models and are defined as follows.

Definition 1 (Probabilistic Circuit)**.** *Given a set of random variables* $\mathbf{X}$*, a* probabilistic circuit *(PC) is based on an acyclic directed graph* $\mathcal{S}$ *containing three types of nodes: tractable* distribution nodes *over a subset of the random variables in* $\mathbf{X}$*, e.g. Gaussian, Categorical, or other exponential families;* sum nodes*, which compute a convex combination (a mixture) of their inputs; and* product nodes*, which compute the product (a factorisation) of their inputs. All leaves of* $\mathcal{S}$ *are distribution nodes and all internal nodes are either sum or product nodes. We assume that* $\mathcal{S}$ *has a single root, which is the output of the PC, computing a density over* $\mathbf{X}$*.*

As mentioned in the introduction, PCs can express many different tractable models (Vergari et al. 2020). Of particular interest in this paper are PCs representing *completely factorised* distributions, $p(\mathbf{x}) = \prod_{d=1}^{D} p(x_d)$, and *Chow-Liu trees* (CLTs) (Chow and Liu 1968), tree-shaped graphical models that can be learnt in polynomial time. CLTs can also be easily converted into PCs (Dang, Vergari, and Broeck

[2]For Monte Carlo, the weights are simply $w(z_i) = \frac{1}{N}$. For Gaussian quadratures the sum of the weights is a function of the domain of integration, but knowledge that $p(\mathbf{x})$ is a probability distribution gives licence to re-normalise the weights in this case.

2020; Di Mauro et al. 2021). We will denote PC structures corresponding to factorised distributions as $\mathcal{S}_{\mathsf{F}}$ and CLT structures as $\mathcal{S}_{\mathsf{CLT}}$. While these structures are arguably simple, we show in the experiment section that continuous mixtures of such PCs outperform all state-of-the-art PC learners on 16 out of 20 common benchmark datasets.

The perhaps most widely known *continuous mixture model* is the *variational autoencoder* (VAE) (Kingma and Welling 2014; Rezende and Mohamed 2015), specifying the model density as $p(\mathbf{x}) = \int p(\mathbf{x} \mid \phi(\mathbf{z}))\, p(\mathbf{z}) \mathrm{d}\mathbf{z}$ where $p(\mathbf{z})$ is an isotropic Gaussian and $p(\mathbf{x} \mid \phi(\mathbf{z}))$ is typically a fully factorised distribution of Gaussians or Binomials, whose parameters are provided by a neural network $\phi(\mathbf{z})$—the so-called decoder—taking $\mathbf{z}$ as input. Since the latent code in VAEs is usually relatively high-dimensional, learning and inference is done via amortised inference (Kingma and Welling 2014).

When using $\mathcal{S}_{\mathsf{F}}$, our models specify in fact the same model as VAEs, which has originally been introduced by McKay (MacKay 1995) under the name *density network*. Our work essentially re-visits McKay's work, who already mentions: *For a hidden vector of sufficiently small dimensionality, a simple Monte Carlo approach to the evaluation of these integrals can be effective.* For this paper, we considered various numerical integration methods and found that randomised quasi-Monte Carlo (RQMC) performs best for our purposes. Moreover, we also use numerical approximation as a *'compilation approach'*, whereby the continuous mixture is converted into a tractable discrete mixture that sets new state of the art among tractable models in a number of datasets.

Similar approaches to our method are HyperSPNs (Shih, Sadigh, and Ermon 2021) and conditional SPNs (Shao et al. 2020), both of which use neural nets to compute the weights of PCs. However, HyperSPNs are primarily a regularisation technique that applies to a single PC, whereas we use neural nets to learn continuous mixtures of PCs. In conditional SPNs the parameters are a function of observed variables, while in this work they are a function of a continuous *latent* space.

Inference and Learning

Our model as specified in (3) consists of a continuous latent space $\mathbf{Z}$ and a given PC structure $\mathcal{S}$, whose parameters $\phi(\mathbf{z})$ are a differentiable function of the latent variables. We will broadly refer to function ϕ as *decoder* and to the model as a whole as *continuous mixtures*. We use $\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$ and $\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$ to denote continuous mixtures with factorised structure and CLT structure, respectively.

Amortised inference (Kingma and Welling 2014; Rezende and Mohamed 2015) is the de-facto standard way to learn continuous mixture models. In this approach, a separate neural network—the so-called *encoder*—represents an approximate posterior $q(\mathbf{z} \mid \mathbf{x})$. The encoder and decoder are learnt simultaneously by maximising the *evidence lower bound* (ELBO)

$$\mathbb{E}_q[\log p(\mathbf{x} \mid \mathbf{z}) - \log q(\mathbf{z} \mid \mathbf{x}) + \log p(\mathbf{z})] \tag{5}$$

which is a lower bound of the (marginal) log-likelihood $\log p(\mathbf{x})$ and thus a principled objective. At the same time, maximising the ELBO is moving q closer to the true posterior in Kullback-Leibler sense, hence tightening the ELBO.

In this paper, we investigate numerical integration as an alternative inference and learning method. In particular, we do not require an encoder or any other parametric form of approximate posterior.

Inference via Numerical Integration

Given some function f, a numerical integration method consists of a set of N integration points $\{\mathbf{z}_i\}_{i=1}^N$ and a weight function $w : \mathbf{z} \mapsto \mathbb{R}$ such that the integration error $\varepsilon = \left| \int f(\mathbf{z})\, \mathrm{d}\mathbf{z} - \sum_{i=1}^N w(\mathbf{z}_i)\, f(\mathbf{z}_i) \right|$ is as small as possible. In this paper, we are interested in approximating the density $p(\mathbf{x}) = \int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, d\mathbf{z}$ of a cm model, so that the integration error with integration points $\{\mathbf{z}_i\}_{i=1}^N$ is given as

$$\varepsilon_{\mathrm{cm}}(\mathbf{x}, \{\mathbf{z}_i\}_{i=1}^N) = \left| \int p(\mathbf{x} \mid \phi(\mathbf{z}))\, p(\mathbf{z})\, d\mathbf{z} - \sum_{i=1}^N w(\mathbf{z}_i)\, p(\mathbf{x} \mid \phi(\mathbf{z}_i)) \right|. \tag{6}$$

Quadrature Rules divide the integration domain into subintervals and approximate the integrand on these intervals with polynomials, which are easy to integrate. They yield a set of deterministic integration points and weights as a function of the degree of the interpolating polynomial. Common quadrature rules like trapezoidal and Simpson's rule achieve error bounds of $\mathcal{O}(N^{-2})$ and $\mathcal{O}(N^{-4})$, respectively. *Gaussian quadrature rules* go a step further and allow us to take into account the distribution of $\mathbf{Z}$; e.g. Gauss-Hermite quadrature is designed for indefinite integrals of the form (1) with $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{1})$. Gaussian quadratures integrate exactly any polynomial of degree $2N-1$ or less, which makes them attractive for general integrands, since by the Weierstrass approximation theorem, any function can be approximated by a polynomial to arbitrary precision under mild regularity conditions (Weierstrass 1885).

Sparse Grids. Unfortunately, quadrature rules do not scale well to high dimensions, since multi-dimensional quadrature rules are usually constructed as the tensor product of univariate rules. If a univariate quadrature rule has an error bound of $\mathcal{O}(N^{-r})$, the corresponding rule in d dimensions with N^d integration points would achieve an error bound $\mathcal{O}(N^{-r/d})$, which degrades quickly due to the curse of dimensionality. Sparse grids (Bungartz and Griebel 2004; Smolyak 1960) try to circumvent that by a special truncation of the tensor product expansion of univariate quadrature rules. This reduces the number of integration points to $O(N(\log N)^{d-1})$ without significant drops in accuracy (Gerstner and Griebel 2010). However, even if the underlying quadrature formulas are positive, sparse grids can yield negative weights $w(\mathbf{z}_i)$. In our preliminary experiments, this property of sparse grids was highly problematic, in particular when inference was used as part of a learning routine.

Monte Carlo (MC) methods cast the integral as an expectation such that we can compute $\int p(\mathbf{x} \mid \mathbf{z})\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}$ as $\mathbb{E}_{p(\mathbf{z})}[p(\mathbf{x} \mid \mathbf{z})] = \sum_{i=1}^N \frac{1}{N} p(\mathbf{x} \mid \mathbf{z}_i)$. MC is easy to use and especially attractive for high-dimensional problems since its convergence rate $\mathcal{O}(N^{-1/2})$ is not directly dependent

on the problem dimensionality. However, this convergence rate decelerates quickly as one increases the number of integration points, which can be too slow. Quasi-Monte Carlo (QMC) methods (Caflisch 1998) replace the (pseudo-)random sequences of integration points of standard MC with low-discrepancy ones, with the intent of reducing the variance of the estimator and converging faster than $\mathcal{O}(N^{-1/2})$. Crucially, QMC is deterministic, which makes it hard to estimate the integration error in practice. Randomised quasi-Monte Carlo (RQMC) reintroduces randomness into the low-discrepancy sequences of integration points, enabling error estimation (via multiple simulations) and essentially turning QMC into a variance reduction method (l'Ecuyer 2016).

In our experiments, we opt for RQMC for two reasons. First, its convergence does not depend directly on the dimensionality, meaning we have more freedom to define the latent space. In fact, we empirically observe that increasing the latent dimensionality does not hurt performance (see additional results in Appendix B). We conjecture that training via numerical integration sufficiently regularises the decoder so that it remains amenable to numerical integration, even when using a relatively large latent dimensionality in the order of tens. Second, in contrast to Monte Carlo, RQMC produces integration points of lower variance (l'Ecuyer 2016), which often facilitates training (see Appendix G); and, in comparison to QMC, RQMC reintroduces randomness into the sets of integration points which helps to avoid overfitting.

Learning the Decoder

In principle, continuous mixture models can be learnt in many ways, such as amortised variational inference (Kingma and Welling 2014) or adversarial training (Goodfellow et al. 2014). However, these methods do not encourage the decoder to be amenable to numerical integration, and thus their approximation by (or compilation to) a mixture of tractable models is subpar. Perhaps not surprisingly, we find that training via numerical integration is the best way to learn and extract expressive mixtures of tractable probabilistic models. We compare numerical integration and variational inference in Figure 1 and Appendix C.

Training via integration simply amounts to selecting a set of integration points $\{\mathbf{z}_i\}_{i=1}^N$ using any numerical integration method of choice and, for some training data $\{\mathbf{x}_j\}_{j=1}^M$, maximising the log-likelihood (LL) of the approximate model with respect to the decoder parameters:

$$LL = \sum_{j=1}^{M} \log \sum_{i=1}^{N} w(\mathbf{z}_i)\, p(\mathbf{x}_j \mid \phi(\mathbf{z}_i)). \tag{7}$$

For $N \to \infty$, this objective converges to the exact log-likelihood of the continuous mixture, and for $1 \ll N < \infty$ it serves as a reasonable approximation.

In particular, when using (RQ)MC methods the inner sum of (7) is unbiased, yielding a *negatively biased* estimate of the true log-likelihood due to Jensen's inequality—i.e. a 'noisy lower bound'—justifying (7) as training objective for similar reasons as the variational ELBO (5). However, (7) should not be confused with the standard ELBO as it does not involve a posterior approximation and, unlike the ELBO, it becomes tight for $N \to \infty$. We further note that (RQ)MC methods to estimate the log-likelihood of latent variable models is not a new idea and has been widely explored either directly (MacKay 1995) or to improve ELBO techniques (Burda, Grosse, and Salakhutdinov 2015; Mnih and Rezende 2016; Buchholz, Wenzel, and Mandt 2018). In this paper, however, we are specifically interested in combining continuous mixtures with *tractable models* via numerical integration, as this direction has been explored rather little.

One can interpret our model in two distinct ways. The first is to interpret it as a 'factory' method, whereby each fixed set of latent variables $\{\mathbf{z}_i\}_{i=1}^N$ yields a tractable model supporting exact likelihood and marginalisation, namely a PC (trivially a mixture of PCs is a PC). The second is to take it as an intractable continuous latent variable model, but one that is amenable to numerical integration. At test time, we are free to choose the set of integration points, possibly changing the integration method and number of integration points N if more or less precision is needed.

Efficient Learning

When using a neural network to fit $p(\mathbf{x} \mid \phi(\mathbf{z}))$, computing the backward pass with respect to the log-likelihood objective in (7) can be memory intensive. We can circumvent that by first finding the K integration points that are most likely to have generated each training instance. We do so via a forward pass (with no gradient computation) that allows us to identify the K values of $\mathbf{z}$ (among the N points $\{\mathbf{z}_i\}_{i=1}^N$ defined by the integration method) that maximise $w(\mathbf{z})\, p(\mathbf{x}_j \mid \phi(\mathbf{z}))$ for each $\mathbf{x}_j$ in a training batch. We then run backprop to optimise a cheaper estimate of the log-likelihood (8) which only requires K values $\{\mathbf{z}_{ij}\}_{i=1}^K$ for each $\mathbf{x}_j$, instead of N

$$LL' = \sum_{j=1}^{M} \log \sum_{i=1}^{K} w(\mathbf{z}_{ij})\, p(\mathbf{x}_j \mid \phi(\mathbf{z}_{ij})). \tag{8}$$

For $K \ll N$ this results in large improvements in memory efficiency. In our experiments, we only use this approximation for non-binary image datasets for which $K = 1$ was already sufficient to get good results.

Latent Optimisation

As mentioned in the last section, once the decoder is trained, we can compile a continuous mixture into a PC by fixing a set of integration points (selected via some integration method), leading to a discrete mixture of PCs. However, rather than using a fixed integration scheme, one might also treat the integration points as 'parameters' and optimise them. More precisely, given training instances $\{\mathbf{x}_j\}_{j=1}^M$, we might find suitable $\{\mathbf{z}_i\}_{i=1}^N$ by maximising the log-likelihood:

$$\arg\max_{\{\mathbf{z}_i\}_{i=1}^N} \sum_{j=1}^{M} \log \sum_{i=1}^{N} w(\mathbf{z}_i)\, p(\mathbf{x}_j \mid \phi(\mathbf{z}_i)). \tag{9}$$

Due to the similarity to (Bojanowski et al. 2018; Park et al. 2019) we refer to this technique as *Latent Optimisation (LO)*. There are, however, a few differences in spirit: in (Bojanowski et al. 2018; Park et al. 2019) the decoder and individual latent

Dataset	BestPC	$\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$	$\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$	$\mathsf{LO}(\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}}))$	Dataset	BestPC	$\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$	$\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$	$\mathsf{LO}(\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}}))$
accid.	**-26.74**	-33.27	-28.69	-28.81	jester	-52.46	**-51.93**	-51.94	-51.94
ad	-16.07	-18.71	-14.76	**-14.42**	kdd	**-2.12**	-2.13	**-2.12**	**-2.12**
baudio	-39.77	**-39.02**	**-39.02**	-39.04	kosarek	-10.60	-10.71	-10.56	**-10.55**
bbc	-248.33	**-240.19**	-242.83	-242.79	msnbc	**-6.03**	-6.14	-6.05	-6.05
bnetflix	-56.27	-55.49	**-55.31**	-55.36	msweb	-9.73	-9.68	-9.62	**-9.60**
book	-33.83	-33.67	-33.75	**-33.55**	nltcs	**-5.99**	**-5.99**	**-5.99**	**-5.99**
c20ng	-151.47	-148.24	**-148.17**	-148.28	plants	-12.54	-12.45	**-12.26**	-12.27
cr52	-83.35	-81.52	**-81.17**	-81.31	pumbs	**-22.40**	-27.67	-23.71	-23.70
cwebkb	-151.84	-150.21	-147.77	**-147.75**	tmovie	-50.81	**-48.69**	-49.23	-49.29
dna	**-79.05**	-95.64	-84.91	-84.58	tretail	-10.84	10.85	-10.82	**-10.81**
Avg. rank	2.85	2.65	1.85	**1.75**					

Table 1: Average test log-likelihoods on 20 density estimation benchmarks. We compare $\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$, $\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$ and $\mathsf{LO}(\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}}))$ with the best performing PC (BestPC) among 5 PC methods: Einets (Peharz et al. 2020a), LearnSPN (Gens and Pedro 2013), ID-SPN (Rooshenas and Lowd 2014), RAT-SPN (Peharz et al. 2020b) and HCLT (Liu and Van den Broeck 2021). For $\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$ and $\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$, we train with 2^{10} integration points and test with a PC compiled with 2^{13} points. LO is run over 2^{10} integration points. See Tables 10 and 11 in Appendix E for more results including standard deviation over 5 random seeds. Higher is better.

representations, one for each training instance, are jointly learnt to minimise the reconstruction error. The latent space is regularised to follow a Gaussian prior, but otherwise is devoid of any probabilistic interpretation.

In our approach, however, the goal is an accurate yet compact approximation to the true continuous mixture model. Training the decoder and integration points together would lead to overfitting, meaning that the decoder would not perform well with different integration points or methods. For that reason, we only optimise the integration points as a *post-processing step*, i.e. the *decoder parameters remain fixed* throughout the optimisation process.

Our LO approach can also be interpreted as a way to learn (or compile) PCs, using continuous mixtures as a teacher model or regularizer. In our experiments, we see that this approach is remarkably effective. LO yields test log-likelihoods similar to that obtained with RQMC while using considerably fewer integration points, and thus delivering smaller PCs.

Experiments

We evaluated our method on common benchmarks for generative models, namely 20 standard density estimation datasets (Lowd and Davis 2010; Van Haaren and Davis 2012; Bekker et al. 2015) as well as 4 image datasets (Binary MNIST (Larochelle and Murray 2011), MNIST (LeCun et al. 1998), Fashion MNIST (Xiao, Rasul, and Vollgraf 2017) and Street View House Numbers (SVHN) (Netzer et al. 2011)). All models were developed in python 3 with PyTorch (Paszke et al. 2019) and trained with standard commercial GPUs. We used RQMC in all experiments ($w(\mathbf{z}_i) = 1/N$). Further experimental details can be found in Appendix A, and our source code is available at github.com/alcorreia/cm-tpm.

Standard Density Estimation Benchmarks

As a first experiment, we compared *continuous mixtures of factorisations*, denoted $\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$, and *continuous mixtures of CLTs*, denoted $\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$, as density estimators on a series of 20 standard commonly used benchmark datasets. In this set of experiments, we fixed the mixing distribution $p(\mathbf{z})$ to a 4-dimensional standard Gaussian and used $N = 2^{10}$ integration points during training. For the decoder we used 6-layer MLPs with LeakyReLUs activations.

At test time, the trained models can be evaluated with any number of integration points N, yielding a mixture of PCs and consequently indeed a standard PC (Vergari et al. 2020). In Table 1 we report the test log-likelihoods for cm models with 2^{13} components, averaged over the 5 random seeds, and for current SOTA PCs. Our results set SOTA log-likelihoods for tractable models on 16 out of 20 datasets and are competitive on the remaining 4. For each dataset, we ranked the performance of the considered models from 1 to 4, and reported the average rank at the bottom of the first half of the table. In particular, we notice a substantial gap in performance between $\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$ and $\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$ for the datasets accidents, ad, dna and pumbs, which are known to be highly structured. We emphasise that we used the exact same hyperparameters for all datasets, and hence our SOTA results do not stem from extensive tuning efforts.

In Figure 1, we plot the performance of our models relative to the best results in Table 1, averaged over all 20 datasets. This shows the effect of the number of integration points at test time and indicates $\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$ generally outperforms $\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$, especially for small numbers of integration points.

Latent Optimisation

Next, we showcase the effect of Latent Optimisation (LO), i.e. learning the integration points after having fit the decoder, as previously discussed. More precisely, we run LO to search for a good set of integration points for a trained $\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$ by maximising (9). We show the results under $\mathsf{LO}(\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}}))$ in Table 1 and Figure 1.

We see that $\mathsf{LO}(\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}}))$ achieves essentially the same performance as $\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$ but with *8 times fewer integration points,* leading to much smaller PCs. However, as can be seen in Figure 1, for a large number of integration points, (RQ)MC estimates already have low-variance and there is little room for improvement with LO. In fact, in this setting

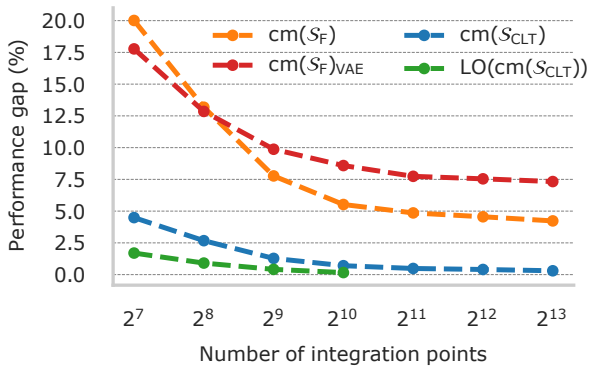

Figure 1: Relative performance gap to the best log-likelihood in Table 1 as a function of the number of integration points at test time and averaged over all 20 datasets. Latent Optimisation is run (on purpose) for fewer number of integration points yet performs best. Lower is better.

LO is prone to overfitting. This can be expected when the number of integration points becomes too large (in comparison to the training data), since they are treated as trainable parameters. For that reason, we limit our LO experiments to 2^{10} integration points in all datasets in Table 1 and 2.

Comparison with Variational Learning

As the main idea of this paper is to relate continuous mixtures to PCs via numerical integration, we used (7) to train our models thus far. However, the training of a continuous mixture model and its subsequent conversion to a PC are orthogonal to each other, and we might also use VAE training to learn continuous mixtures. That is, we can train a standard VAE with a *small latent dimension*, maximising the ELBO (5) with amortised variational inference, and then convert the resulting model into a PC using RQMC or LO.

We evaluated this alternative with the same experimental setup and 20 density estimation datasets. We learnt continuous mixtures with VAE training (Kingma and Welling 2014) and subsequently numerically integrated the resulting model, denoted $\mathrm{cm}(\mathcal{S}_\mathsf{F})_\mathsf{VAE}$, with RQMC. Note that we used exactly the same architecture for both $\mathrm{cm}(\mathcal{S}_\mathsf{F})$ and $\mathrm{cm}(\mathcal{S}_\mathsf{F})_\mathsf{VAE}$, with the only difference being the training method. Figure 1 shows that $\mathrm{cm}(\mathcal{S}_\mathsf{F})_\mathsf{VAE}$ is outperformed by $\mathrm{cm}(\mathcal{S}_\mathsf{F})$, even for large numbers of integration points at test time. We offer two explanations for this result: First, it might be that $\mathrm{cm}(\mathcal{S}_\mathsf{F})_\mathsf{VAE}$ models are less amenable to our numerical approximation techniques and that their true log-likelihood is actually higher, or conversely, that models *trained* with numerical integration are more amenable to numerical integration at test time. Second, it might also be that numerical integration leads to better model training for *small latent dimensionality*. We provide affirmative evidence for the latter in Appendix C. However, evidently, VAE training is superior for large latent dimensionality, as numerical integration degrades quickly in high dimensional spaces. See Appendix C for more comprehensive experimental details and further results.

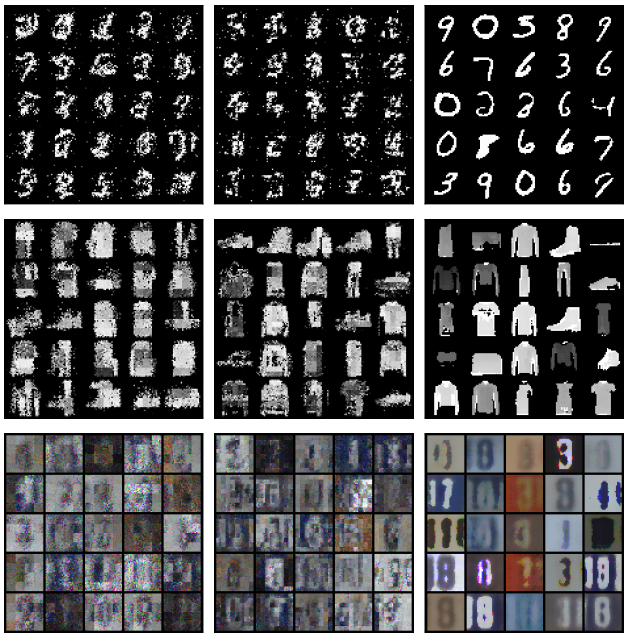

Figure 2: Samples from 'Small Einet' (left column), 'Big Einet' (middle column) and $\mathrm{cm}(\mathcal{S}_\mathsf{F})$ (right column).

Binary MNIST

We also evaluated our models on Binary MNIST (Larochelle and Murray 2011). We followed the same experimental protocol as in the previous experiments, except that we employed a larger latent dimensionality of 16 and increased the number of integration points during training to 2^{14}. We did *not* use convolutions and stuck to 6-layer MLPs. We ran $\mathrm{cm}(\mathcal{S}_\mathsf{F})$ and $\mathrm{cm}(\mathcal{S}_\mathsf{CLT})$ and applied LO to both final models for up to 50 epochs, using early stopping on the validation set to avoid overfitting. Table 2 shows that $\mathrm{cm}(\mathcal{S}_\mathsf{CLT})$ outperforms $\mathrm{cm}(\mathcal{S}_\mathsf{F})$ overall and that LO is remarkably effective when few integration points are used. In Table 3, we compare our models against Einets (Peharz et al. 2020a), large scale PCs designed to take advantage of GPU accelerators. We considered Einets of different sizes[3] and used Poon-Domingos architectures (Poon and Domingos 2011), which recursively partition the image into contiguous square blocks.

Image Datasets

For non-binary image data we used a convolutional architecture similar to that of DCGAN (Radford, Metz, and Chintala 2015) but also included residual blocks as in (Van Den Oord, Vinyals et al. 2017). For both MNIST and SVHN data, we used the same architecture and trained $\mathrm{cm}(\mathcal{S}_\mathsf{F})$ models with 16 latent dimensions and $K=1$ (see Efficient Learning). Once more, we compared against Einets of different sizes. For both Einets and our models, pixels were modelled with 256-dimensional categorical distributions[4]. In all cases, we use no

[3]For Binary MNIST, 'Small Einet' and 'Big Einet' had respectively 5 and 84 million parameters; for MNIST, 11 and 90 million parameters; and for SVHN, 28 and 186 million parameters.

[4]See Appendix E for results using Normal distributions instead.

Model	N. Param	Number of integration points at test time					
		2^7	2^8	2^9	2^{10}	2^{12}	2^{14}
$\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$ (LO)	1.2M	-167.29 (-144.00)	-150.67 (-135.89)	-138.55 (-129.15)	-129.24 (-123.44)	-116.42	-108.69
$\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$ (LO)	4.8M	-127.59 (-114.02)	-119.09 (-110.02)	-113.15 (-107.14)	-108.30 (-104.37)	-101.55	-97.48

Table 2: Binary MNIST test log-likelihoods for $\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$ and $\mathrm{cm}(\mathcal{S}_{\mathsf{CLT}})$ trained with 2^{14} integration points. In parentheses we report test log-likelihoods obtained via latent optimisation. Higher is better.

Model	'Small Einet'	'Big Einet'	Ours (2^{14})
Binary MNIST	0.206	0.184	**0.179**
MNIST	1.490	1.415	**1.282**
Fashion-MNIST	3.938	3.737	**3.546**
SVHN	6.442	**5.961**	6.307

Table 3: Bits per dim. (bpd) for image data. Lower is better.

auxiliary clustering algorithm to assign datapoints to components of a sum node. Such a pre-processing step is applicable to any method and does not add to the analysis in this paper. That is why sample quality in Figure 2 is worse than that reported in (Peharz et al. 2020a), where images were clustered and a dedicated Einet was trained on each cluster.

As seen in Table 3, continuous mixtures outperform Einets in all image datasets but SVHN. That is remarkable since our models are extremely compact with the decoder given by a light convolutional architecture of approximately 100K free parameters for MNIST data, and 300K for SVHN; orders of magnitude smaller than the competing Einets. Moreover, our models also achieve better sample quality. In Figure 2, we see samples from $\mathrm{cm}(\mathcal{S}_{\mathsf{F}})$ are clearly sharper and do not suffer from intense pixelation like those from Einets.

Other Tractable Queries

Our models also support efficient marginalisation, since the discrete approximation obtained via numerical integration is a PC in itself. That allows us to handle missing data and perform tasks like inpainting out-of-the-box, without any extra modelling steps. While this is not the focus of the paper—these queries are well-established for PCs, and it is not surprising that our models support them as well—we do present a couple of interesting experiments in Appendix F. We successfully trained our model on MNIST with substantial parts of the data missing. Note that such a training procedure is delicate for intractable models like VAEs. Furthermore, we included inpainting experiments on Binary MNIST, MNIST and Fashion MNIST, i.e. reconstructing missing data at test time, using a model trained on complete data.

Discussion

Our experiments show that continuous mixtures of PCs (or actually their discrete approximations yielding again PCs) outperform most previous PC methods on several datasets. At first this might appear surprising. For fixed N, a discrete mixture with N components is at least as expressive as a continuous mixture approximated by N integration points, since the former has mixture components with free (private) parameters, while the latter has components which are determined via a shared neural network, and thus entangled in a complex way. Moreover, the PCs of previous works have been deeper and used more sophisticated architectures than our continuous mixtures. A comparison between our models and discrete mixtures with the same shallow structures is deferred to Appendix D.

The main reason for the efficacy of our approach might be the continuity of the neural network, which topologically relates the latent and observable space, thus identifying some underlying latent structure; this is in fact one of the attractive and widely appreciated properties of VAEs. Yet, the effect of continuity on generalisation has not been much studied, and our results provide an interesting pointer in this regard. Why does continuity promote generalisation, or act as some form of regularisation? For the one, there might be an Occam's razor effect at work, since our models are usually much smaller in terms of free parameters, yet they are expressive due to the non-linear nature of neural nets. Furthermore, dependence among components introduced via the latent space might effectively facilitate learning by avoiding redundant or 'dead' components, which have been observed in vanilla PCs (Dang, Liu, and Van den Broeck 2022).

These results have two important consequences for future work on tractable probabilistic models: (i) continuous latent spaces seem to be a valuable tool for learning tractable models and (ii) PCs in general seem to have untouched potential not yet exploited by existing learning methods.

Conclusion

In this paper, we have investigated the marriage of continuous mixtures and tractable probabilistic models. We have observed that, even with simple structures and standard numerical integration methods, continuous latent variables facilitate the learning of expressive PCs, as confirmed by SOTA results on many datasets. Moreover, we have proposed latent optimisation as an effective way to derive competitive mixture models with relatively few components (integration points). We believe continuous mixtures are a promising tool for learning tractable probabilistic models as well as developing new hybrid inference models (Tan and Peharz 2019).

Our model is not without limitations, however. In particular, numerical integration is a computationally expensive training approach, and we assume fixed PC structures (independent of the latent variables) that have to be defined or learnt a priori. These two issues are promising avenues for future work, especially with extensions to more complex structures, like HCLTs (Liu and Van den Broeck 2021).

Acknowledgements

We thank the Eindhoven Artificial Intelligence Systems Institute (EAISI) for its support. This research was supported by the Graz Center for Machine Learning (GraML). This work was partially funded by the EU European Defence Fund Project KOIOS (EDF-2021-DIGIT-R-FL-KOIOS). We also thank Guy Van den Broeck and team for their feedback that helped us improve the paper.

References

Bekker, J.; Davis, J.; Choi, A.; Darwiche, A.; and Van den Broeck, G. 2015. Tractable learning for complex probability queries. *Advances in Neural Information Processing Systems*, 28.

Bojanowski, P.; Joulin, A.; Lopez-Pas, D.; and Szlam, A. 2018. Optimizing the Latent Space of Generative Networks. In *International Conference on Machine Learning (ICML)*, 600–609. PMLR.

Buchholz, A.; Wenzel, F.; and Mandt, S. 2018. Quasi-monte carlo variational inference. In *International Conference on Machine Learning (ICML)*, 668–677. PMLR.

Bungartz, H.-J.; and Griebel, M. 2004. Sparse grids. *Acta numerica*, 13: 147–269.

Burda, Y.; Grosse, R.; and Salakhutdinov, R. 2015. Importance weighted autoencoders. *arXiv preprint arXiv:1509.00519*.

Caflisch, R. E. 1998. Monte carlo and quasi-monte carlo methods. *Acta numerica*, 7: 1–49.

Chow, C.; and Liu, C. 1968. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory*, 14(3): 462–467.

Correia, A.; Peharz, R.; and de Campos, C. P. 2020. Joints in Random Forests. *Advances in Neural Information Processing Systems*, 33.

Dang, M.; Liu, A.; and Van den Broeck, G. 2022. Sparse Probabilistic Circuits via Pruning and Growing. In *The 5th Workshop on Tractable Probabilistic Modeling*.

Dang, M.; Vergari, A.; and Broeck, G. V. d. 2020. Strudel: Learning Structured-Decomposable Probabilistic Circuits. *arXiv preprint arXiv:2007.09331*.

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

Di Mauro, N.; Gala, G.; Iannotta, M.; and Basile, T. M. 2021. Random probabilistic circuits. In *Uncertainty in Artificial Intelligence*, 1682–1691. PMLR.

Gens, R.; and Pedro, D. 2013. Learning the structure of sum-product networks. In *International Conference on Machine Learning (ICML)*, 873–880. PMLR.

Gerstner, T.; and Griebel, M. 2010. Sparse Grids. In Cont, R., ed., *Encyclopedia of Quantitative Finance*. John Wiley and Sons.

Ghahramani, Z. 2015. Probabilistic machine learning and artificial intelligence. *Nature*, 521(7553): 452–459.

Goodfellow, I. J.; Pouget-Abadie, J.; Mirza, M.; Xu, B.; Warde-Farley, D.; Ozair, S.; Courville, A. C.; and Bengio, Y. 2014. Generative Adversarial Nets. In *NIPS*.

Jaynes, E. T. 2003. *Probability Theory: The Logic of Science*. Cambridge University Press.

Kingma, D. P.; and Welling, M. 2014. Auto-Encoding Variational Bayes. In *International Conference on Learning Representations (ICLR)*. ArXiv:1312.6114.

Kisa, D.; Van den Broeck, G.; Choi, A.; and Darwiche, A. 2014. Probabilistic sentential decision diagrams. In *Proceedings of the 14th international conference on principles of knowledge representation and reasoning (KR)*, 1–10.

Larochelle, H.; and Murray, I. 2011. The neural autoregressive distribution estimator. In *Proceedings of the fourteenth international conference on artificial intelligence and statistics*, 29–37. JMLR Workshop and Conference Proceedings.

LeCun, Y.; Bottou, L.; Bengio, Y.; and Haffner, P. 1998. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324.

Liu, A.; and Van den Broeck, G. 2021. Tractable regularization of probabilistic circuits. *Advances in Neural Information Processing Systems*, 34.

Lowd, D.; and Davis, J. 2010. Learning Markov network structure with decision trees. In *2010 IEEE International Conference on Data Mining*, 334–343. IEEE.

l'Ecuyer, P. 2016. Randomized quasi-Monte Carlo: An introduction for practitioners. In *International Conference on Monte Carlo and Quasi-Monte Carlo Methods in Scientific Computing*, 29–52. Springer.

MacKay, D. J. 1995. Bayesian neural networks and density networks. *Nuclear Instruments and Methods in Physics Research Section A: Accelerators, Spectrometers, Detectors and Associated Equipment*, 354(1): 73–80.

Mnih, A.; and Rezende, D. 2016. Variational inference for monte carlo objectives. In *International Conference on Machine Learning (ICML)*, 2188–2196.

Netzer, Y.; Wang, T.; Coates, A.; Bissacco, A.; Wu, B.; and Ng, A. Y. 2011. Reading digits in natural images with unsupervised feature learning. In *NIPS Workshop on Deep Learning and Unsupervised Feature Learning*.

Park, J. J.; Florence, P.; Straub, J.; Newcombe, R.; and Lovegrove, S. 2019. Deepsdf: Learning continuous signed distance functions for shape representation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 165–174.

Paszke, A.; Gross, S.; Massa, F.; Lerer, A.; Bradbury, J.; Chanan, G.; Killeen, T.; Lin, Z.; Gimelshein, N.; Antiga, L.; et al. 2019. Pytorch: An imperative style, high-performance deep learning library. *arXiv preprint arXiv:1912.01703*.

Peharz, R. 2015. *Foundations of Sum-Product Networks for Probabilistic Modeling*. Ph.D. thesis, Graz University of Technology.

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*, 39(10): 2030–2044.

Peharz, R.; Lang, S.; Vergari, A.; Stelzner, K.; Molina, A.; Trapp, M.; Van den Broeck, G.; Kersting, K.; and Ghahramani, Z. 2020a. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *International Conference on Machine Learning (ICML)*, 7563–7574. PMLR.

Peharz, R.; Vergari, A.; Stelzner, K.; Molina, A.; Shao, X.; Trapp, M.; Kersting, K.; and Ghahramani, Z. 2020b. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Uncertainty in Artificial Intelligence*, 334–344. PMLR.

Poon, H.; and Domingos, P. 2011. Sum-product networks: A new deep architecture. In *2011 IEEE International Conference on Computer Vision Workshops (ICCV Workshops)*, 689–690. IEEE.

Radford, A.; Metz, L.; and Chintala, S. 2015. Unsupervised representation learning with deep convolutional generative adversarial networks. *arXiv preprint arXiv:1511.06434*.

Rahman, T.; Kothalkar, P.; and Gogate, V. 2014. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of Chow-Liu trees. In *Joint European conference on machine learning and knowledge discovery in databases*, 630–645. Springer.

Rezende, D.; and Mohamed, S. 2015. Variational inference with normalizing flows. In *International Conference on Machine Learning (ICML)*, 1530–1538. PMLR.

Rooshenas, A.; and Lowd, D. 2014. Learning sum-product networks with direct and indirect variable interactions. In *International Conference on Machine Learning (ICML)*, 710–718. PMLR.

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*, 401–412. PMLR.

Shih, A.; Sadigh, D.; and Ermon, S. 2021. HyperSPNs: Compact and Expressive Probabilistic Circuits. *Advances in Neural Information Processing Systems*, 34.

Smolyak, S. A. 1960. Interpolation and quadrature formulas for the classes W_s^α and E_s^α. In *Doklady Akademii Nauk*, volume 131, 1028–1031. Russian Academy of Sciences.

Tan, P. L.; and Peharz, R. 2019. Hierarchical Decompositional Mixtures of Variational Autoencoders. In Chaudhuri, K.; and Salakhutdinov, R., eds., *Proceedings of the 36th International Conference on Machine Learning (ICML)*, volume 97 of *Proceedings of Machine Learning Research*, 6115–6124. PMLR.

Tomczak, J.; and Welling, M. 2018. VAE with a VampPrior. In *International Conference on Artificial Intelligence and Statistics*, 1214–1223. PMLR.

Van Den Oord, A.; Vinyals, O.; et al. 2017. Neural discrete representation learning. *Advances in neural information processing systems*, 30.

Van Haaren, J.; and Davis, J. 2012. Markov network structure learning: A randomized feature generation approach. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 26.

Vergari, A.; Choi, Y.; Peharz, R.; and Van den Broeck, G. 2020. Probabilistic Circuits: Representations, Inference, Learning and Applications. http://starai.cs.ucla.edu/slides/AAAI20.pdf. Tutorial at AAAI 2020.

Weierstrass, K. 1885. Über die analytische Darstellbarkeit sogenannter willkürlicher Functionen einer reellen Veränderlichen. *Sitzungsberichte der Königlich Preußischen Akademie der Wissenschaften zu Berlin*, 2: 633–639.

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

Zhao, H.; Poupart, P.; and Gordon, G. 2016. A unified approach for learning the parameters of sum-product networks. *arXiv preprint arXiv:1601.00318*.