Hierarchical Decompositional Mixtures of Variational Autoencoders

Ping Liang Tan [1 2] **Robert Peharz** [1]

Abstract

Variational autoencoders (VAEs) have received considerable attention, since they allow us to learn expressive neural density estimators effectively and efficiently. However, learning and inference in VAEs is still problematic due to the sensitive interplay between the generative model and the inference network. Since these problems become generally more severe in high dimensions, we propose a novel hierarchical mixture model over low-dimensional VAE experts. Our model decomposes the overall learning problem into many smaller problems, which are coordinated by the hierarchical mixture, represented by a sum-product network. In experiments we show that our models outperform classical VAEs on almost all of our experimental benchmarks. Moreover, we show that our model is highly data efficient and degrades very gracefully in extremely low data regimes.

1. Introduction

Accurately estimating the probability distribution of high-dimensional data is a central and long standing goal in unsupervised learning. Given an accurate joint model over the random variables of interest allows us to answer – in principle – any statistical question we might have about our data, such as inferring (conditional) marginal distributions, computing expectations, imputing missing data, and sampling new data instances.

Variational Autoencoders (VAEs) (Kingma & Welling, 2014) provide a particularly flexible approach to density estimation. The VAE model defines an infinite mixture $\int p(\mathbf{x} \mid f(\mathbf{z};\theta))\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}$, where prior $p(\mathbf{z})$ is typically isotropic Gaussian, f is a neural network parametrized by θ, and $p(\mathbf{x} \mid f(\mathbf{z};\theta))$ is some "simple" distribution, e.g. independent Gaussian, Bernoulli, etc. Classically, this model has been known as *density network* (MacKay, 1995), and inference was performed using Monte Carlo methods, which do not scale well to large datasets. Kingma & Welling (2014) improved the applicability of this model by using an *inference network* to amortize the inference effort over data instances (Gershman & Goodman, 2014), applying *stochastic variational inference* (SVI) for optimizing the evidence lower bound (ELBO) (Paisley et al., 2012; Hoffman et al., 2013), and employing the *re-parametrization trick* (Titsias & Gredilla-Lázaro, 2014; Rezende et al., 2014) to reduce the variance in gradient estimates of the ELBO.

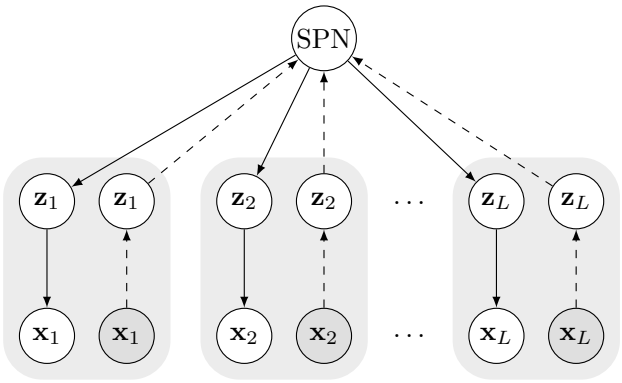


Figure 1. Our proposed hierarchical decompositional mixture over variational autoencoders, a divide-and-conquer scheme which ameliorates learning and inference in comparison to classical VAEs. The overall variable scope $\mathbf{x}$ is split into L sub-scopes $\mathbf{x}_1, \ldots \mathbf{x}_L$, following the structure of a sum-product network (SPN). Each sub-scope is modeled by an expert, represented by a variational autoencoder (VAE), which constitute the leaves of the SPN. In the **generative process** (depicted with solid lines), the SPN probabilistically selects a combination of VAE experts, each of which generates its part of the scope. The **inference process** (depicted with dashed lines) utilizes local inference networks for each VAE, yielding local estimates for the evidence-lower-bound (ELBO). The local estimates are fed into the SPN, yielding an ELBO estimate for the entire model (Theorem 2). See supplementary for an extended version of this figure.

However, inference and learning in VAEs are still problematic. For the one, the additional approximation introduced by amortization is generally compensated by the generative model, leading to deteriorated learning (Cremer et al., 2018).

[1]Department of Engineering, University of Cambridge, UK [2]DSO National Laboratories, Singapore. Correspondence to: Ping Liang Tan <plt28@gmail.com>, Robert Peharz <rp587@cam.ac.uk>.

Proceedings of the 36^{th} International Conference on Machine Learning, Long Beach, California, PMLR 97, 2019.

On the other hand, improving inference quality, e.g. by using more samples in *importance weighted autoencoders* (IWAEs) (Burda et al., 2016), might impair the learning signal for the inference network (Rainforth et al., 2018). These problems with learning VAEs can be expected to become harder for data of higher dimensionality.

Therefore, in this paper, we propose a natural *divide-and-conquer* approach and propose a novel structured density estimator, formulated as a hierarchical decompositional mixture over VAEs. The hierarchical mixture part of our model is represented by a sum-product network (SPN) (Poon & Domingos, 2011). SPNs naturally implement the desired divide-and-conquer approach, by using mixtures (sum nodes) and factorization (product nodes) in a recursive and nested structure, captured by a directed computational graph. The inputs (leaves) to this graph are distribution functions over a (typically small) sub-set of the model variables. A major advantage of SPNs is, that they permit a large variety of tractable inference routines. For example, evaluating the represented density or *any of its sub-marginals* can be done *exactly and efficiently*, provided that the leaf distributions allow the corresponding operations.

The key observation used in this paper is that SPNs allow *arbitrary* representations for the leaves – also intractable ones like VAE distributions. The idea of this hybrid model class, which we denote as sum-product VAE (SPVAE), is to combine the best of both worlds: On the one hand, it extends the model capabilities of SPNs by using flexible VAEs as leaves; one the other hand, it applies the above mentioned divide-and-conquer approach to VAEs, which can be expected to lead to an easier learning problem. Since SPNs can be interpreted as structured latent variable models (Zhao et al., 2015; Peharz et al., 2017), the generative process of SPVAEs can be depicted as in Fig. 1 (solid lines).

While defining an SPVAE distribution imposes no theoretical problem, it is not immediately clear how to perform inference and learning in these models. Fortunately, as shown in this paper, SPNs and VAEs play very well together: In Theorem 1 we show that SPN inputs – normally required to be probabilities – can be replaced with local ELBOs (i.e. over the variable scope captured by the leaf), yielding an ELBO of the overall SPVAE model at the output. Furthermore, in Theorem 2, we show that the exact local ELBOs can be replaced with Monte Carlo estimates obtained from *leaf-local* inference networks, allowing us to perform stochastic variational inference in SPVAEs. Thus, the inference process in SPVAEs decomposes in the same way as the generative process, as depicted in Fig. 1 (dashed lines). Please see also the supplementary for a more detailed version of Fig. 1.

We compare SPVAEs with classical VAEs on the MNIST, CIFAR-10, and SVHN image datasets. For each dataset, we consider two versions: i) interpreting images as continuous signals and using Gaussians for the VAE outputs, and ii) interpreting images as discrete data on $0 \dots 255$ and using Binomial distributions as VAE outputs. On all of these 6 benchmarks, SPVAEs clearly outperform classical VAEs in terms of test likelihood (estimated with 5000 importance-weighted samples). At the same time, due to their decompositional nature, SPVAE models are almost an order of magnitude smaller than VAEs. Moreover, we show that SPVAEs are more data efficient than VAEs: on all benchmarks we can reduce the amount of training data down to $\approx 10\%$, without significantly deteriorating the test performance. Even for extremely low data regimes, SPVAEs degrade much more gracefully than VAEs.

The paper is organized as follows. In Section 2 we discuss the required background and related work. In Section 3 we introduce our hybrid model and discuss learning and inference. Section 4 presents our experiments and Section 5 concludes the paper. Proofs are deferred to Section 6.

2. Background and Related Work

Let $\mathbf{x} = (x_1, \dots, x_D)$ be a collection of random variables. Our goal is to estimate the density $p(\mathbf{x})$ (subsuming probability mass functions w.r.t. to some counting measure) of $\mathbf{x}$ using a dataset of samples drawn i.i.d. from the underlying distribution. In the following we discuss several approaches relevant for this paper.

Density Networks A *density network* (DN) (MacKay, 1995) offers a flexible blackbox density estimator for high dimensional data. In a DN, we postulate a K-dimensional latent vector (code) $\mathbf{z}$ with isotropic Gaussian prior $\mathbf{z} \sim \mathcal{N}(\mathbf{0}, \mathbf{I})$. The latent code is passed through a neural network f_θ parameterized by θ. The outputs of the neural network parametrize some "simple" distribution (e.g. independent Normal, Bernoulli, etc) over the data $\mathbf{x}$, i.e. θ specifies a conditional distribution $p_\theta(\mathbf{x} \,|\, \mathbf{z}) = p(\mathbf{x} \,|\, f_\theta(\mathbf{z}))$. Consequently, the modeled data distribution is $p(\mathbf{x}) = \int p(\mathbf{x} \,|\, f_\theta(\mathbf{z}))\, p(\mathbf{z})\, \mathrm{d}\mathbf{z}$. When f_θ is replaced with a linear function, we recover traditional latent factor analysis. Thus, DNs with a generic non-linear neural network yield a form of non-linear latent factor analysis. Posterior inference and learning in DNs is notoriously hard, and the Monte Carlo inference methods proposed in (MacKay, 1995) do not scale well to high-dimensional data.

Variational Autoencoders DNs have been re-discovered in (Kingma & Welling, 2014), but were referred to as *probabilistic decoder*. Kingma & Welling introduced several techniques in order to make inference practical: i) they introduced a *second DN* parametrized by ϕ, called the *inference network* or *probabilistic encoder*, representing an

approximate posterior $q_\phi(\mathbf{z} \mid \mathbf{x})$; ii) they train encoder and decoder simultaneously by optimizing the *evidence-lower-bound* (ELBO)

$$\mathcal{L} = \mathbb{E}_{q_\phi} \left[\log p_\theta(\mathbf{x}, \mathbf{z}) - \log q_\phi(\mathbf{z} \mid \mathbf{x}) \right], \tag{1}$$

using stochastic optimization (Paisley et al., 2012; Hoffman et al., 2013); iii) since gradient estimates of (1) have a large variance, they re-parametrize $\mathbf{z}$ according to $\mathbf{z} = g(\epsilon)$, where g is differentiable and ϵ is a helper random variable with some fixed distribution. This is commonly known as the re-parametrization trick (Rezende et al., 2014; Titsias & Gredilla-Lázaro, 2014; Schulman et al., 2015). The combination of decoder (i.e. the generative DN) and encoder (i.e. the inference network) is called *variational autoencoder* (VAE). Since a VAE is actually a combination of two networks, also its application is typically twofold in practice: On the one hand, the encoder can be used to embed the data $\mathbf{x}$ into a latent space, while on the other hand, we might use VAEs for density estimation (i.e. the decoder). This paper puts emphasis on the latter. Thus, for the remainder of the paper, we will refer to both DNs and VAEs simply as VAEs, if a distinction does not matter.

Importance Weighted VAEs Burda et al. (2016) pointed out that the common assumption of a completely factorized posterior approximation imposes a strong assumption on VAE learning. As a remedy, they propose an *importance weighted* ELBO estimator

$$\hat{\mathcal{L}}_k = \log \frac{1}{k} \sum_{i=1}^{k} \frac{p_\theta(\mathbf{x}, \mathbf{z}_i)}{q_\phi(\mathbf{z}_i \mid \mathbf{x})}, \tag{2}$$

where $\mathbf{z}_i$ are independent draws from the inference network. For $k'' > k'$ we have $\mathbb{E}[\hat{\mathcal{L}}_{k''}] \geq \mathbb{E}[\hat{\mathcal{L}}_{k'}]$, i.e. the importance weighted ELBO becomes increasingly tighter with larger k, and for $k \to \infty$, $\hat{\mathcal{L}}_k$ converges to the exact log-likelihood $\log p(x)$. VAEs trained with (2) are usually denoted as *importance weighted VAEs* (IWAEs).

Sum-Product Networks Sum-product networks (SPNs) (Poon & Domingos, 2011) are a deep generalization of mixture models. They model a high-dimensional distribution using a recursive definition, utilizing the building blocks of i) *mixtures*, i.e. convex combinations of distributions (i.e. a non-negative linear combination with weights summing to one), and ii) *factorization*. The structure of this recursive arrangement is captured by an acyclic directed graph $\mathcal{G}$, whose internal nodes represent the sum and product operations, each taken over the children in the graph. We assume in this paper that $\mathcal{G}$ is connected and has a single root, which represents the distribution over $\mathbf{x}$ modeled by the SPN.

The SPN structure $\mathcal{G}$ must obey two rules, namely *completeness*, meaning that mixtures are taken only over distributions with the same variable scope,[1] and *decomposability*, meaning that products are taken only over distributions with non-overlapping scope. Consequently, product nodes effectively split larger scopes into non-overlapping sub-scopes, i.e. they implement a divide-and-conquer scheme. The basis of this recursive scheme are user-provided distributions (over certain sub-scopes), which are represented by the leaves of the SPN. Commonly used leaf distributions are parametric forms such as Gaussian, Poisson, categorical, etc (Molina et al., 2017; Vergari et al., 2019).

Since the sum nodes represent mixture models, SPNs can be naturally interpreted as hierarchical latent variable models (Zhao et al., 2015; Peharz et al., 2017). In particular, the generative process in SPNs can be described as a hierarchical decision process, governed by the joint state of all sum nodes. This process selects a set of leaves, whose scope is a partitioning of the overall scope $\mathbf{x}$. An exact sample is then produced by simply concatenating samples from the selected leaves.

Further Related Work In (Mocanu & Mocanu, 2018), so-called mixtures of VAEs have been considered, which are, however, no proper mixtures in a probabilistic sense (which our model subsume). This work also does not discuss density estimation but rather considers an ensemble of VAEs for classification tasks and one-shot learning. Goyal et al. (2017) propose non-parametric VAEs for hierarchical representation learning. The hierarchical organization, however, is restricted to the prior over latent code $\mathbf{z}$, while still a standard VAE decoder is used. This does not yield a model with ameliorated inference, as in our case. Similarly, both (Tomczak & Welling, 2018) and (Dilokthanakul et al., 2016) introduce mixture priors to VAEs, but do not apply a divide-and-conquer modeling approach, as in our case. Ranganath et al. (2016) consider structured variational families, with the aim to increase expressivity of the variational distribution and to structure the computational aspect of inference. This approach is only very coarsely related to ours, as we propose a novel hybrid model combining a tractable model (SPNs) with an intractable one (VAEs). *Ladder VAEs* (Sønderby et al., 2016) have as goal, similar as our paper, to improve the density estimation of VAEs. To this end, they propose a modified inference procedure, which is, however, tailored to multiple stochastic layers, and requires dataset dependent tuning of the variational objective. Ladder VAEs could also be incorporated in our approach, but for simplicity and generality we employ IWAEs in this paper. Johnson et al. (2016) combined graphical models and VAEs; their focus, however, lies on incorporating domain knowledge into structured models, rather than improving density estimation. Recently, Stelzner et al. (2019) also proposed a hybrid system combining SPNs and VAE-like inference

[1] In SPN jargon, *scope* denotes a subset of $\mathbf{x}$.

networks. Their approach, however, is diametrical to ours, as they use SPNs *within* an intractable model, while we use SPN *over* intractable models.

Furthermore, SPNs have been equipped with other expressive models. The approach most similar in spirit to ours is perhaps (Trapp et al., 2018), which uses an SPN structure to combine *Gaussian process experts*. This model, however, puts into focus conditional models rather than density estimation, and still uses tractable models as SPN leaves. Our approach, however, is the first which uses expressive intractable models as SPN leaves, which arguably extends the model capabilities of SPN-based models.

3. Sum-Product Variational Autoencoders

In this section, we introduce our hybrid model combining SPNs and VAEs. We start with a description of the generative process of our model.

3.1. Generative Model

In order to define SPVAEs, recall that SPNs are in fact a "glue language" for specifying probability distributions. Assume any given structure $\mathcal{G}$ of an SPN, where internal nodes are either sums or products, and leaves are distribution functions over some sub-scope of the overall collection of random variables $\mathbf{x}$. Furthermore, assume that the scope $\mathrm{sc}(n) \subseteq \mathbf{x}$ has been assigned to all nodes $n \in \mathcal{G}$, and that $\mathcal{G}$ satisfies the completeness and decomposability conditions, cf. (Poon & Domingos, 2011) and Section 2.

Let $L_\mathcal{G}$ be set of all leaves in $\mathcal{G}$. Then the formalism of SPNs guarantees that any choice of distribution functions p_l, where $l \in L_\mathcal{G}$, leads to a proper distribution function over the whole scope $\mathbf{x}$. In this paper, we opt for VAEs (in fact DNs) as SPN leaf distributions, i.e. each leaf distribution is of the form

$$p_l(\mathbf{x}_l) = \int p_{\theta_l}(\mathbf{x}_l \mid \mathbf{z}_l)\, p(\mathbf{z}_l)\, \mathrm{d}\mathbf{z}_l, \tag{3}$$

where $\mathbf{x}_l$ is shorthand for $\mathrm{sc}(l)$, and $\mathbf{z}_l$ and θ_l are *private* latent VAE codes and VAE decoder parameters for leaf l.

We can analyze the overall SPVAE distribution as follows. Recall, that every sum node n in an SPN computes a convex combination of its children:

$$p_n(\mathbf{x}_n) = \sum_{c \in \mathrm{ch}(n)} w_{nc}\, p_c(\mathbf{x}_c), \tag{4}$$

where $\mathrm{ch}(n)$ denotes the children of n, and w_{nc} are the sum-weights, satisfying $\sum_{c \in \mathrm{ch}(n)} w_{nc} = 1$, $w_{nc} \geq 0, \forall c \in \mathrm{ch}(n)$. As discussed in (Zhao et al., 2015; Peharz et al., 2017), each sum node n is naturally associated with a latent variable y_n, whose states correspond to the sum's children, and which selects a state with probability w_{nc}. Let $\mathbf{y}$ be the collection of all y_n for all sum nodes $n \in \mathcal{G}$. Each assignment for $\mathbf{y}$ selects a sub-structure of $\mathcal{G}$, a so-called *induced tree* $\mathcal{T}$ (Zhao et al., 2016). In particular, an induced tree $\mathcal{T}$ is obtained from $\mathcal{G}$ by removing, for each sum node n, all children which do not correspond to the state y_n, i.e. only the child corresponding to y_n is kept. Furthermore, all nodes and edges which cannot be reached from the root, are deleted.

As shown in (Zhao et al., 2016), the SPN distribution can always be written as

$$p(\mathbf{x}) = \sum_{\mathcal{T} \sim \mathcal{G}} \prod_{(n,c) \in \mathcal{T}} w_{nc} \prod_{l \in L_\mathcal{T}} p_l(\mathbf{x}_l), \tag{5}$$

where the sum in (5) runs over all possible induced trees obtainable from $\mathcal{G}$. Note that $\prod_{(n,c)\in\mathcal{T}} w_{nc}$ is the selection probability for tree $\mathcal{T}$. Moreover, one can show that the scopes of the leaves $L_\mathcal{T}$ always are a proper partition of $\mathbf{x}$, thus (5) is a proper mixture distribution over $\mathbf{x}$. The number of represented components is the number of induced trees, which grows *exponentially* in the depth of the SPN.

Plugging the VAE leaves (3) into (5), we yield a compact expression for the overall SPVAE distribution:

$$p_{SP}(\mathbf{x}) = \sum_{\mathcal{T} \sim \mathcal{G}} \prod_{(n,c) \in \mathcal{T}} w_{nc} \prod_{l \in L_\mathcal{T}} \int p_{\theta_l}(\mathbf{x}_l \mid \mathbf{z}_l)\, p(\mathbf{z}_l)\, \mathrm{d}\mathbf{z}_l. \tag{6}$$

Expression (6) highlights that SPVAEs combine two very distinct types of mixture models in a hierarchical way: The latent SPN variables $\mathbf{y}$ select a set of VAE leaves (via the induced tree), which subsequently produce a sample over their local scope via their generative process $\mathbf{z}_l \to \mathbf{x}_l$. The overall generative process of an SPVAE is illustrated in Fig. 1 (solid lines) and in the supplementary.

3.2. Inference and Learning

If we could evaluate the integrals in (6) exactly, feeding them into the SPN would yield an exact evaluation of the SPVAE distribution at the SPN's root. By taking gradients of this evaluation we could perform maximum-likelihood learning w.r.t. to the SPVAEs parameters, i.e. all sum-weights and the parameters of leaf VAEs. Unfortunately, SPVAEs clearly inherit the intractability of VAEs, i.e. marginalizing the $\mathbf{z}_l$'s is hard and needs to be approximated.

A natural approach to perform inference in SPVAEs is to follow the compositional structure of the generative process also for inference. Consequently, we define for each VAE leaf (3) a corresponding *private inference network* $q_{\phi_l}(\mathbf{z}_l \mid \mathbf{x}_l)$. Using this collection of inference networks, we can replace each leaf distribution (3) with a lower bound,

and define

$$\mathcal{L}_{SP}(\mathbf{x}) = \log \left[\sum_{\mathcal{T} \sim \mathcal{G}} \prod_{(n,c) \in \mathcal{T}} w_{nc} \prod_{l \in L_{\mathcal{T}}} e^{\mathcal{L}_l(\mathbf{x}_l)} \right], \text{ where} \tag{7}$$

$$\mathcal{L}_l(\mathbf{x}_l) = \mathbb{E}_{q_{\phi_l}}[\log p_{\theta_l}(\mathbf{x}_l, \mathbf{z}_l) - \log q_{\phi_l}(\mathbf{z}_l \mid \mathbf{x}_l)]. \tag{8}$$

Note that (7) is the log of the SPN's output, when each VAE leaf has been replaced by its corresponding exponentiated local ELBO, restricted the the leaf's scope. It turns out, that $\mathcal{L}_{SP}(\mathbf{x})$ is an ELBO for SPVAEs.

Theorem 1. *Let an SPVAE with SPN structure $\mathcal{G}$, sum-weights $\boldsymbol{w}$, and VAE leaves with decoder and encoder parameters θ_l, ϕ_l, for $l \in L_{\mathcal{G}}$ be given. Then $\mathcal{L}_{SP}(\mathbf{x}) \leq \log p_{SP}(\mathbf{x})$.*

The proof is provided in Section 6. In some sense, Theorem 1 states that "an SPN over leaf ELBOs is an ELBO for the SPVAE distribution" (being slightly imprecise with log and exp). Consequently, we could learn SPVAEs by optimizing (7) jointly over sum-weights and all VAE parameters. Since the leaf-ELBOs $\mathcal{L}_l(\mathbf{x}_l)$ are hard to compute, we would replace them with Monte Carlo estimates, as e.g. (2), and apply stochastic optimization (Hoffman et al., 2013).

However, it is not immediately clear, whether using estimates for the leaf-ELBOs is a valid approach, since (7) is a non-linear function, and a non-linear function of unbiased estimators is generally biased. Moreover, since (7) is actually a large $logsumexp$ operation over leaf-ELBOs, we would introduce a *positive bias*, since $logsumexp$ is convex. Thus, optimizing a lower bound with an unknown positive bias is not well justified.

Fortunately, it turns out that SPNs are benign in this regard, and play well together with the IWAE ELBO.

Theorem 2. *Let an SPVAE with SPN structure $\mathcal{G}$, sum-weights $\boldsymbol{w}$, and VAE leaves with decoder and encoder parameters θ_l, ϕ_l, for $l \in L_{\mathcal{G}}$ be given, and define*

$$\hat{\mathcal{L}}_{SP,k}(\mathbf{x}) = \log \left[\sum_{\mathcal{T} \sim \mathcal{G}} \prod_{(n,c) \in \mathcal{T}} w_{nc} \prod_{l \in L_{\mathcal{T}}} e^{\hat{\mathcal{L}}_{l,k}(\mathbf{x}_l)} \right], \tag{9}$$

where $\hat{\mathcal{L}}_{l,k}(\mathbf{x}_l)$ is the k-sample IWAE estimator for leaf l, cf. (2). Then $\hat{\mathcal{L}}_{SP,k}(\mathbf{x})$ is an unbiased estimator for some lower bound of $\log p_{SP}(\mathbf{x})$.

The proof is provided in Section 6. Theorem 2 gives us license for a simple implementation of variational learning, illustrated in Algorithm 1.

3.3. SPVAE Structure

While Algorithm 1 allows us to learn the model parameters $\boldsymbol{w}$, $\theta_{1:L}$ and the inference parameters $\phi_{1:L}$ simultaneously and end-to-end, we were assuming that the SPN structure $\mathcal{G}$ and scope assignments $\mathrm{sc}(n)$ for each node n are already given. Structure learning is an actively researched field in SPNs, and various strategies have been proposed to obtain $\mathcal{G}$ in a data-driven way, as for example top-down co-clustering (Dennis & Ventura, 2012; Gens & Domingos, 2013; Rooshenas & Lowd, 2014; Vergari et al., 2015; Molina et al., 2018), bottom up learning based on the information bottleneck (Peharz et al., 2013), hard EM (Kalra et al., 2018), and graph partitioning (Jaini et al., 2018). Also, using random SPN structures has been recently considered in (Peharz et al., 2018). Each of these algorithms can be used, possibly with some adaption, to provide an SPN structure for Algorithm 1.

Algorithm 1 Stochastic Variational EM for SPVAE

input SPN structure $\mathcal{G}$ with $L = |L_{\mathcal{G}}|$ leaves
  initialize sum weights $\boldsymbol{w}$
  initalize VAE parameters $\theta_{1:L}$, $\phi_{1:L}$
  while stopping criterion is false **do**
    $\hat{\mathcal{L}} \leftarrow 0$
    for $\mathbf{x}$ in minibatch **do**
      for $l \in L_{\mathcal{G}}$ **do**
        draw k samples $\mathbf{z}_{l,1} \dots \mathbf{z}_{l,k}$ from $q_{\phi_l}(\mathbf{z}_l \mid \mathbf{x}_l)$
        $\hat{\mathcal{L}}_l \leftarrow \log \frac{1}{k} \sum_{i=1}^{k} \frac{p_{\theta_l}(\mathbf{x}_l, \mathbf{z}_{l,i})}{q_{\phi_l}(\mathbf{z}_{l,i} \mid \mathbf{x}_l)}$
      end for
      $\hat{\mathcal{L}}_{SP,k} \leftarrow$ SPN evaluation with $e^{\hat{\mathcal{L}}_l}$ instead of $p_l(\mathbf{x}_l)$
      $\hat{\mathcal{L}} \leftarrow \hat{\mathcal{L}} + \hat{\mathcal{L}}_{SP,k}$
    end for
    gradient ascent step with $\nabla_{\boldsymbol{w},\theta_{1:L},\phi_{1:L}} \hat{\mathcal{L}}$ (autodiff)
  end while
output $\boldsymbol{w}, \theta_{1:L}, \phi_{1:L}$

In this paper, we focus on image datasets, and leverage an SPN structure tailored to images, as proposed in Poon & Domingos (2011). This algorithm starts with a $D \times D$ image, and recursively considers all possible axis-aligned splits with a certain step-size Δ. This recursive splitting process stops when the dimensions of an obtained sub-image falls below $\Delta \times \Delta$. We denote the rectangles which are not split further as *leaf regions*, and the other rectangles as *internal regions*. Each leaf region is equipped with K_{leaf} VAEs and each internal region with K_{sum} sum nodes – the corresponding region for each of these nodes represents their scope in the final SPVAE. The root region, the original $D \times D$ image, is equipped with only 1 sum, which represents the SPVAE's output. Finally, for each possible axis-aligned split, we take all cross-products of distributions contained in the sub-rectangles, and connect them as children of the sums in the parent rectangle. This construction scheme is guaranteed to deliver a complete and decomposable SPN (Poon & Domingos, 2011). Note that for larger Δ, the

SPN part becomes larger and the scope modeled by the VAE leaves smaller. Thus, in our experiments we select a reasonable choice for Δ, in order to balance the SPN and VAE parts. Please see the supplementary for a small but complete example of the employed structure.

4. Experiments

We performed experiments on three common visual datasets,[2] namely MNIST (LeCun et al.), containing 28×28 images of hand-written images, Street View House Numbers (SVHN) (Netzer et al., 2011), containing 32×32 images of house numbers, and CIFAR-10 (Krizhevsky, 2009), containing 32×32 images of natural objects. For our experiments, SVHN and CIFAR-10 were converted to 8-bit gray images. For each dataset, we considered two modeling approaches, namely i) interpreting pixels as continuous values on the interval $[0, 1]$, and ii) interpreting pixels as discrete values in $\{0, \dots, 255\}$.

The following setting was applied to all VAEs, both for classical VAEs and SPVAE leaves. When interpreting pixels as continuous, all VAE decoders return pixel-wise means and standard deviations, parameterizing Gaussian distributions. When interpreting pixels as discrete, all VAE decoders returned a success probability for each pixel, parameterizing Binomials with $N = 255$. All decoders and encoders were implemented using multilayer perceptions (MLP) with softplus activation. During training, we consistently used 5 importance weighted samples for ELBO estimates (2).

The SPN part of our models was constructed by applying the Poon-Domingos structure (Poon & Domingos, 2011), see also Section 3.3, where we used $\Delta = 7$ for MNIST and $\Delta = 8$ for SVHN and CIFAR-10. That is, for all datasets, the SPN dictates a recursive split of the overall image into a 4×4 array of super-pixels, where the super-pixel size is 7×7 for MNIST and 8×8 for SVHN and CIFAR-10.

For simplicity and in oder to demonstrate the efficacy of the SPVAE approach, we used $K_{\text{sum}} = 2$ sum nodes per internal region and $K_{\text{leaf}} = 2$ VAEs per leaf region (super-pixel). Thus, in total we have 32 VAEs which are governed by the SPN. In particular, since the SPN must essentially select one VAE per super-pixel, this instance of SPVAEs can be understood as a mixture of $2^{16} = 65536$ components, each component being a product of 16 VAE distributions. For numerical stability, all SPN operations were transformed into the log-domain, i.e. the inputs were regular ELBO estimates (rather than exponentiated ones), products were replaced with sums, and sums with log-sum-exp operations. Sum parameters were also represented in the log-domain and their normalization was ensured by subtracting their log-sum-exp. We implemented all models in Tensorflow (Abadi M. et al., 2015) and used Adam (Kingma & Ba, 2015) with its default parameters for optimizing the respective ELBOs. We used a batch size of 128 throughout all our experiments.

[2]Code available under https://github.com/cambridge-mlg/SPVAE.

Table 1. Performance on test set, 5000-sample IWAE ELBO

	Continuous			Discrete		
	mnist	svhn	cifar	mnist	svhn	cifar
SPVAE	**2819**	**1936**	**1283**	**-1532**	**-3891**	**-5543**
VAE	2598	1442	896	-2351	-4965	-7200
Conv-SPVAE	2702	**2101**	**1397**	**-927**	**-3666**	**-4562**
Conv-VAE	**2907**	1896	1191	-2099	-4115	-6752

Table 2. Model size (in thousands of parameter) of best performing VAEs and SP-VAEs

model	Continuous			Discrete		
	mnist	svhn	cifar	mnist	svhn	cifar
SPVAE	**413**	**511**	**511**	**310**	**461**	**353**
VAE	1545	1713	1653	1158	3108	3288
Conv-SPVAE	1259	1604	**1028**	1257	1603	**1026**
Conv-VAE	**894**	**1219**	1219	**894**	**1436**	1219

The quality of density estimation in VAEs and SPVAEs depends both on the model size and the dimensionality of the latent codes. Thus, we treated the number of hidden units H per neural network layer and the dimensionality n_z of latent VAE codes as hyper-parameters, and cross-validated them on a validation set. For MNIST, we randomly selected $10k$ images from the training set for the validation set; for CIFAR-10, the $60k$ images were randomly divided into sets of sizes $40k$ / $10k$ / $10k$, which were used as training, validation, and test sets. For SVHN, we used the first 26032 images from the extra set as validation set (i.e. of the same size as the test set). The same H was used for each layer (decoder and encoder), and in the case of SPVAEs, for each VAE leaf. In order to keep the sizes of the overall models comparable, we used ranges $n_z \in \{1, 2, 5, 25, 50, 100, 150, 200\}$, $H \in \{30, 100, 300, 600\}$ for VAEs, and $n_z \in \{1, 2, 5, 25, 50\}$, $H \in \{8, 16, 32\}$ for SPVAEs.

No regularization was applied, but we used early stopping in order to prevent overfitting. In particular, we evaluated the training progress on the validation set every 128 batches and stop training if the performance on the validation set decreased five times consecutively. After early stopping, we restore the model to the point of best performance on the validation set and report the log-likelihood on the test set, estimated using a 5000-sample IWAE ELBO.

Results are presented in Table 1. We see that SPVAEs clearly outperform classical VAEs on all 6 benchmarks, often by a considerable margin. Moreover, as shown in Table 2, the model sizes for SPVAEs are almost an order of magnitude smaller than for VAEs. This stems from the fact, that rather small MLPs were used in the SPVAE leaves, and that the

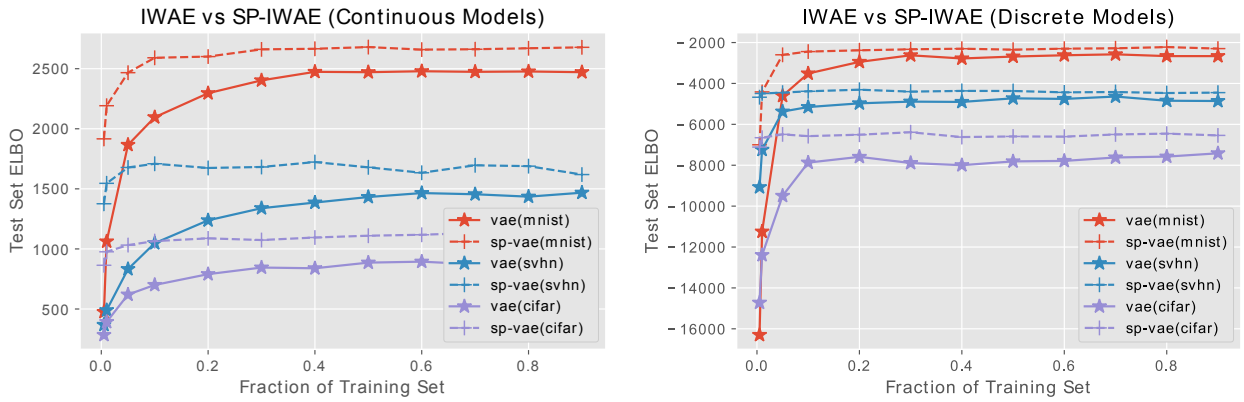


Figure 2. Degradation of test ELBO as training set size is reduced. Dashed lines are for SPVAEs, solid lines are for VAEs. Left: results for continuous models (conditional Gaussian). Right: results for discrete models (conditional Binomial).

weigth matrices grow quadratically in the number of hidden units. In order to make sure, that the better performance of SPVAEs does not stem from using a "deeper" model, we also trained VAEs with 3 hidden layers. This, however, yielded worse results than for 2-layer VAEs. These results speak clearly in favor for SPVAEs, and the divide-and-conquer approach to density estimation they implement.

A potential caveat might be that the used SPN structures implement a convolution-like operation, and that the improved performance stems from this additional structure. Therefore, we repeated our experiments, but where all VAEs were replaced with convolutional VAEs (Conv-VAEs). Additionally, since also our models can be equipped with ConvNets, we also tried SPVAEs equipped with convolutional VAE leaves (Conv-SPVAEs). For Conv-VAEs, we used two 3×3 convolutional layers with 32 and 64 filters, respectively, applying stride 2×2, followed by a fully connected layer. We cross-validated the number of units in the fully connected layer in $\{32, 64, 128\}$ and $n_z \in \{1, 2, 5, 25, 50, 100, 150, 200\}$. For Conv-SPVAEs, we used two 3×3 convolutional layers with 4 and 8 filters, respectively, applying stride 1×1 (due to the small scopes of VAE leaves), and also followed by a fully connected layer. We cross-validated the number of units in the fully connected layer in $\{8, 32, 64\}$ and $n_z \in \{1, 2, 5, 25, 50\}$, for each leaf.

The results for the convolutional variants are shown in the lower segments of Tables 1 and 2. We see, that convolutional structures help both Conv-VAEs and Conv-SPVAEs. Generally, our hybrid models still improve over regular Conv-VAEs, except on continuous mnist, where Conv-SPVAEs are even worse than SPVAEs. This outlier result might stem from an unfortunate parameter intialization.

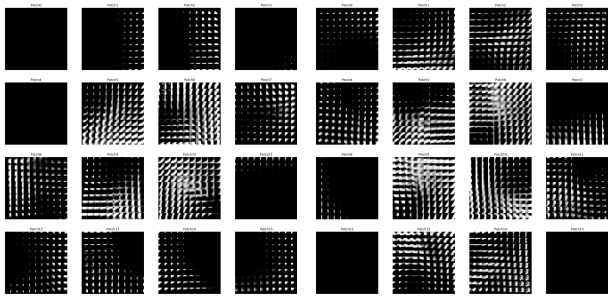
4.1. Low Data Regime

Furthermore, we investigated data efficiency in learning SPVAEs and VAEs. Reliably estimating a density in many dimensions becomes significantly harder compared to low dimensional problems, a problem frequently coined as the "curse of dimensionality". Since SPVAEs decompose an overall density estimation problem into smaller "orchestrated" ones, we could also expect that they learn in a more data efficient way. To this end, we artificially reduce the amount of data in the training set, and compare the SPVAE and VAE performances. We randomly downsampled the training set, reducing its size first linearly in steps of 10% from 100% to 10% and then exponentially from 10% to 0.5% in 3 steps. Both SPVAE and VAE were trained on the same reduced training sets. The hyperparameters and latent dimensions for each model were cross-validated, as in the experiments over the whole datasets.

In Figure 2, we see that SPVAEs are indeed dramatically more data efficient than VAEs. As the training set is reduced, we see that VAE's performance on the test set remains relatively stable until some point, beyond which its performance decreases at an accelerating rate. The same phenomenon happens for SPVAEs, except that they consistently outperform standard VAEs, remain stable for a greater reduction in training set size (until $\approx 10\%$). Furthermore, at the most extreme reduction of dataset size, they degrade by a smaller amount from their peak performance.

4.2. Latent Space of SPVAEs

Finally, VAEs allow to retrieve a latent embedding of the modeled data via their inference network. SPVAEs can be used in a similar way, by retrieving latent embeddings for both the leaf VAEs and the SPN, by interpreting the latent variable semantic in SPNs as probabilistic encoder,



(a) 'Left' VAE (b) 'Right' VAE

Figure 3. Latent space of the 32 SPVAE leaves trained on continuous MNIST. Each image in the 4×4 array is a latent space representation, akin to (Kingma & Welling, 2014). The 4×4 array is arranged in the same way as the super-pixels in our SPN structure. The assignment 'Left' and 'Right' VAE is arbitrary.

as explored in (Vergari et al., 2018). To give some insights into the learned latent space of SPVAEs, we illustrate a 2-dimensional representation for each VAE leaf trained on MNIST (continuous) in Fig. 3. This embedding was retrieved as in (Kingma & Welling, 2014), but for each VAE leaf individually. We see that each VAE leaf learns a reasonable latent embedding, which is distinct for the two alternative VAEs in each leaf. Each of the VAEs specializes to a certain sub-cluster of features occurring its assigned super-pixel.

5. Conclusion

We presented SPVAEs, a novel structured model which combines a tractable model (SPNs) and an intractable model (VAEs) in a natural way. As shown in our experiments, this leads to i) better density estimates, ii) smaller models, and iii) improved data efficiency when compared to classical VAEs. Future work includes more extensive experiments with SPN structures and VAE variants. Furthermore, the decompositional structure of SPVAEs naturally lends itself towards distributed model learning and inference. We also observed in our experiments that SPVAEs allowed larger learning rates than VAEs (up to 0.1) during training. Investigating this effect is also subject for future research. A current downside is that the decomposed nature of SPVAEs cause run-times approximately a factor 5 slower than for VAEs. However, the execution time SPVAEs could be reduced by more elaborate designs, facilitating a higher degree of parallelism, for example using vectorization and distributed learning.

More generally, and in some sense complementary to the results in (Stelzner et al., 2019), this paper demonstrates that combining tractable probabilistic models and expressive intractable models is a promising avenue for future research.

6. Proofs

Proof for Theorem 1. Each term $\mathcal{L}_l(\mathbf{x}_l)$ in (7) is the standard ELBO for leaf l. Thus $e^{\mathcal{L}_l(\mathbf{x}_l)} \leq p_l(\mathbf{x}_l)$, where $p_l(\mathbf{x}_l)$ is the leaf distribution defined in (3). Due to non-negativity of $e^{\mathcal{L}_l(\mathbf{x}_l)}$, it holds that $\prod_{l \in L_\mathcal{T}} e^{\mathcal{L}_l(\mathbf{x}_l)} \leq \prod_{l \in L_\mathcal{T}} p_l(\mathbf{x}_l)$, and moreover

$$\sum_{\mathcal{T} \sim \mathcal{G}(n,c) \in \mathcal{T}} \prod w_{nc} \prod_{l \in L_\mathcal{T}} e^{\mathcal{L}_l(\mathbf{x}_l)} \leq \sum_{\mathcal{T} \sim \mathcal{G}(n,c) \in \mathcal{T}} \prod w_{nc} \prod_{l \in L_\mathcal{T}} p_l(\mathbf{x}_l),$$

where the right hand side is the SPVAE distribution (5), (6). The inequality holds since the mixture $\sum_{\mathcal{T} \sim \mathcal{G}} \prod_{(n,c) \in \mathcal{T}} w_{nc} \ldots$, applied to both sides, is a non-negative combination. Taking the $\log$ completes the proof. □

Proof for Theorem 2. Recall that the k-sample IWAE estimator for leaf l is given as $\hat{\mathcal{L}}_{l,k}(\mathbf{x}_l) = \log \frac{1}{k} \sum_{i=1}^{k} \frac{p_\theta(\mathbf{x}_l, \mathbf{z}_{l,i})}{q_\phi(\mathbf{z}_{l,i} \mid \mathbf{x}_l)}$, where $\mathbf{z}_l = (\mathbf{z}_{l,1}, \ldots, \mathbf{z}_{l,k})$ are i.i.d. samples from q_{ϕ_l}. Let $L = |L_\mathcal{G}|$ be the number of leaves in the SPN and $\mathbf{z}_{1:L}$ contain the $\mathbf{z}_l$'s for all leaves. The randomness of $\hat{\mathcal{L}}_{SP,k}(\mathbf{x})$ stems entirely from $\mathbf{z}_{1:L}$, i.e. its expectation is given as $\mathbb{E}_{\mathbf{z}_{1:L}}\Big[\log \Big[\sum_{\mathcal{T} \sim \mathcal{G}} \prod_{(n,c) \in \mathcal{T}} w_{nc} \prod_{l \in L_\mathcal{T}} e^{\hat{\mathcal{L}}_{l,k}(\mathbf{x}_l)}\Big]\Big]$. Now, we have

$$\mathbb{E}[\hat{\mathcal{L}}_{SP,k}] \leq \log \left[\mathbb{E}_{\mathbf{z}_{1:L}} \left[\sum_{\mathcal{T} \sim \mathcal{G}(n,c) \in \mathcal{T}} \prod w_{nc} \prod_{l \in L_\mathcal{T}} e^{\hat{\mathcal{L}}_{l,k}(\mathbf{x}_l)} \right] \right] \tag{10}$$

$$= \log \left[\sum_{\mathcal{T} \sim \mathcal{G}(n,c) \in \mathcal{T}} \prod w_{nc} \prod_{l \in L_\mathcal{T}} \mathbb{E}_{\mathbf{z}_l} \left[e^{\hat{\mathcal{L}}_{l,k}(\mathbf{x}_l)} \right] \right] \tag{11}$$

$$= \log \left[\sum_{\mathcal{T} \sim \mathcal{G}(n,c) \in \mathcal{T}} \prod w_{nc} \prod_{l \in L_\mathcal{T}} p_l(\mathbf{x}_l) \right] \tag{12}$$

$$= \log p_{SP}(\mathbf{x}). \tag{13}$$

In (10) we apply Jensen's inequality. In (11) the expectation $\mathbb{E}_{\mathbf{z}_{1:L}}$ is distributed over leaves. To see why this is a valid operation, first note that $\mathbb{E}_{\mathbf{z}_{1:L}}$ clearly commutes with the sum over $\mathcal{T}$ and the product over w_{nc}, as they do not depend on $\mathbf{z}_{1:L}$. Furthermore, note that all $\mathbf{z}_l$ are drawn independently from each other, such that $\mathbb{E}_{\mathbf{z}_{1:L}}$ can be written as nested expectations, each over one $\mathbf{z}_l$. Since each factor in the innermost product depends on exactly one $\mathbf{z}_l$, we an distribute the expectations. In (12), we have $\mathbb{E}_{\mathbf{z}_l}\left[e^{\hat{\mathcal{L}}_{l,k}(\mathbf{x}_l)}\right] = p_l(\mathbf{x}_l)$, since $e^{\hat{\mathcal{L}}_{l,k}(\mathbf{x}_l)} = \frac{1}{k} \sum_{i=1}^{k} \frac{p_\theta(\mathbf{x}_l, \mathbf{z}_{l,i})}{q_\phi(\mathbf{z}_{l,i} \mid \mathbf{x}_l)}$, which is an unbiased estimator for $p_l(\mathbf{x}_l)$. Finally, (13) holds by definition (3) and (6), which completes the proof. □

Acknowledgements

We want to thank Zoubin Ghahramani for his "historical remarks" on density networks and variational autoencoders. PLT acknowledges the DSTA Scholarship which sponsored his studies. RP: This project has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 797223 — HYBSPN.

References

Abadi M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. URL https://www.tensorflow.org/. Software available from tensorflow.org.

Burda, Y., Grosse, R., and Salakhutdinov, R. Importance weighted autoencoders. In *ICLR, https://arxiv.org/abs/1509.00519*, 2016.

Cremer, C., Li, X., and Duvenaud, D. Inference suboptimality in variational autoencoders. In *Proceedings ICML*, volume 80, pp. 1078–1086, 2018.

Dennis, A. and Ventura, D. Learning the architecture of sum-product networks using clustering on variables. In *Proceedings of NIPS*, pp. 2042–2050, 2012.

Dilokthanakul, N., Mediano, P. A. M., Garnelo, M., Lee, M. C. H., Salimbeni, H., Arulkumaran, K., and Shanahan, M. Deep unsupervised clustering with gaussian mixture variational autoencoders. *arXiv preprint arXiv:1611.02648*, 2016.

Gens, R. and Domingos, P. Learning the structure of sum-product networks. *Proceedings of ICML*, pp. 873–880, 2013.

Gershman, S. J. and Goodman, N. D. Amortized inference in probabilistic reasoning. In *Proceedings of CogSci*, 2014.

Goyal, P., Hu, Z., Liang, X., Wang, C., and Xing, E. P. Nonparametric variational auto-encoders for hierarchical representation learning. In *Proceedings of ICCV*, pp. 5094–5102, 2017.

Hoffman, M., Blei, D., Wang, C., and Paisley, J. Stochastic variational inference. *Journal of Machine Learning Research*, 14:1303–1347, 2013.

Jaini, P., Ghose, A., and Poupart, P. Prometheus : Directly learning acyclic directed graph structures for sum-product networks. In *Proceedings of PGM*, pp. 181–192, 2018.

Johnson, M., Duvenaud, D. K., Wiltschko, A., Adams, R. P., and Datta, S. R. Composing graphical models with neural networks for structured representations and fast inference. In *Proceedings of NIPS*, pp. 2946–2954, 2016.

Kalra, A., Rashwan, A., Hsu, W., Poupart, P., Doshi, P., and Trimponias, G. Online structure learning for feed-forward and recurrent sum-product networks. In *Proceedings of NIPS*, pp. 6944–6954. 2018.

Kingma, D. P. and Ba, J. Adam: A method for stochastic optimization. In *Proceedings of ICLR*, 2015.

Kingma, D. P. and Welling, M. Auto-encoding variational Bayes. In *ICLR*, 2014. arXiv:1312.6114.

Krizhevsky, A. Learning multiple layers of features from tiny images (tech report), 2009. https://www.cs.toronto.edu/~kriz/cifar.html.

LeCun, Y., Cortes, C., and Burges, C. J. C. The MNIST database of handwritten digits. http://yann.lecun.com/exdb/mnist/.

MacKay, D. J. C. 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, 1995.

Mocanu, D. C. and Mocanu, E. One-shot learning using mixture of variational autoencoders: a generalization learning approach. *arXiv preprint arXiv:1804.07645*, 2018.

Molina, A., Natarajan, S., and Kersting, K. Poisson sum-product networks: A deep architecture for tractable multivariate Poisson distributions. In *Proceedings of AAAI*, 2017.

Molina, A., Vergari, A., Di Mauro, N., Natarajan, S., Esposito, F., and Kersting, K. Mixed sum-product networks: A deep architecture for hybrid domains. In *Proceedings of AAAI*, 2018.

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

Paisley, J., Blei, D. M., and Jordan, M. I. Variational Bayesian inference with stochastic search. In *Proceedings of ICML*, pp. 1367–1374, 2012.

Peharz, R., Geiger, B., and Pernkopf, F. Greedy part-wise learning of sum-product networks. In *Proceedings of ECML/PKDD*, pp. 612–627. Springer Berlin, 2013.

Peharz, R., Gens, R., Pernkopf, F., and Domingos, P. On the latent variable interpretation in sum-product networks. *IEEE transactions on pattern analysis and machine intelligence*, 39(10):2030–2044, 2017.

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Kersting, K., and Ghahramani, Z. Probabilistic deep learning using random sum-product networks. In *UAI 2018 Workshop on Uncertainty in Deep Learning (UDL); arXiv preprint arXiv:1806.01910*. 2018.

Poon, H. and Domingos, P. Sum-product networks: A new deep architecture. In *Proceedings of UAI*, pp. 337–346, 2011.

Rainforth, T., Kosiorek, A., Le, T. A., Maddison, C., Igl, M., Wood, F., and Teh, Y. W. Tighter variational bounds are not necessarily better. In *Proceedings of ICML*, volume 80, pp. 4277–4285, 2018.

Ranganath, R., Tran, D., and Blei, D. Hierarchical variational models. In *Proceedings of ICML*, pp. 324–333, 2016.

Rezende, D. J., Mohamed, S., and Wierstra, D. Stochastic backpropagation and approximate inference in deep generative models. In *Proceedings of ICML*, pp. 1278–1286, 2014.

Rooshenas, A. and Lowd, D. Learning Sum-Product Networks with Direct and Indirect Variable Interactions. In *Proceedings of ICML*, pp. 710–718, 2014.

Schulman, J., Heess, N., Weber, T., and Abbeel, P. Gradient estimation using stochastic computation graphs. In *Proceedings of NIPS*, pp. 3528–3536, 2015.

Sønderby, C. K., Raiko, T., Maaløe, L., Sønderby, S. K., and Winther, O. Ladder variational autoencoders. In *Proceedings of NIPS*, pp. 3738–3746, 2016.

Stelzner, K., Peharz, R., and Kersting, K. Faster attend-infer-repeat with tractable probabilistic models. In *Proceedings of ICML*, 2019.

Titsias, M. K. and Gredilla-Lázaro, M. Doubly stochastic variational Bayes for non-conjugate inference. In *Proceedings of ICML*, pp. 1971–1979, 2014.

Tomczak, J. and Welling, M. Vae with a vampprior. In *Proceedings of AISTATS*, pp. 1214–1223, 2018.

Trapp, M., Peharz, R., Rasmussen, C. E., and Pernkopf, F. Learning deep mixtures of gaussian process experts using sum-product networks. *arXiv preprint arXiv:1809.04400*, 2018.

Vergari, A., Di Mauro, N., and Esposito, F. Simplifying, regularizing and strengthening sum-product network structure learning. In *Proceedings of ECML/PKDD*, pp. 343–358. Springer, 2015.

Vergari, A., Peharz, R., Di Mauro, N., Molina, A., Kersting, K., and Esposito, F. Sum-product autoencoding: Encoding and decoding representations using sum-product networks. In *Proceedings of AAAI*, 2018.

Vergari, A., Molina, A., Peharz, R., Ghahramani, Z., Kersting, K., and Valera, I. Automatic Bayesian density analysis. In *Proceedings of AAAI*, 2019.

Zhao, H., Melibari, M., and Poupart, P. On the relationship between sum-product networks and Bayesian networks. In *Proceedings of ICML*, pp. 116–124, 2015.

Zhao, H., Adel, T., Gordon, G., and Amos, B. Collapsed variational inference for sum-product networks. In *Proceedings of ICML*, pp. 1310–1318, 2016.