Random Sum-Product Networks: A Simple but Effective Approach to Probabilistic Deep Learning

Robert Peharz[1], **Antonio Vergari**[2], **Karl Stelzner**[3], **Alejandro Molina**[3], **Xiaoting Shao**[3], **Martin Trapp**[4]
Kristian Kersting[3], **Zoubin Ghahramani**[1,5]
University of Cambridge[1], MPI for Intelligent Systems[2], TU Darmstadt[3], TU Graz[4], Uber AI Labs[5]
{rp587,zoubin}@cam.ac.uk, antonio.vergari@tue.mpg.de,
{stelzner, molina,xiaoting.shao, kersting}@cs.tu-darmstadt.de, martin.trapp@tugraz.at

Abstract

Sum-product networks (SPNs) are expressive probabilistic models with a rich set of exact and efficient inference routines. However, in order to guarantee exact inference, they require specific structural constraints, which complicate learning SPNs from data. Thereby, most SPN structure learners proposed so far are tedious to tune, do not scale easily, and are not easily integrated with deep learning frameworks. In this paper, we follow a simple "deep learning" approach, by generating unspecialized random structures, scalable to millions of parameters, and subsequently applying GPU-based optimization. Somewhat surprisingly, our models often perform on par with state-of-the-art SPN structure learners and deep neural networks on a diverse range of generative and discriminative scenarios. At the same time, our models yield well-calibrated uncertainties, and stand out among most deep generative and discriminative models in being robust to missing features and being able to detect anomalies.

1 INTRODUCTION

Intelligent systems should both be able to deal with uncertain inputs, as well as express uncertainties over their outputs. Especially the latter is a crucial point in automatic decision-making processes, such as medical diagnosis and planning systems for autonomous agents. Therefore, it is no surprise that probabilistic approaches have recently gained great momentum in deep learning, which has led to a variety of probabilistic models such as variational autoencoders (VAEs) [45, 28], generative adversarial nets (GANs) [24], neural auto-regressive density estimators (ARDEs) [29, 52, 51], and normalizing flows (NFs) [18, 27].

However, most of these probabilistic deep learning systems have limited capabilities when it comes to *inference*. First, they have to resort to approximate inference in most inference scenarios, e.g., marginalization and conditioning for ARDEs, NFs. Moreover, some models do not allow to evaluate the likelihood, either because they lack of a probability density (e.g. GANs) or evaluating it is intractable (e.g. VAEs). Furthermore, even when tractable approximations can be carried out, there is no guarantee that these computations yield a calibrated estimation of the underlying uncertainty in data, or even conform to human expectations [10, 34].

In this landscape, *sum-product networks* (SPNs) [11, 41] are a promising avenue, as they are a class of deep probabilistic models permitting *exact* and *efficient* inference. In particular, SPNs are able to compute *any* marginalization and conditioning query in time linear of the model's representation size. This property is a hallmark of SPNs, distinguishing them from the other probabilistic models mentioned above. Nevertheless, despite their attractive inference properties, SPNs have received comparatively limited attention in the deep learning community. A major reason for this is that the structure of an SPN needs to obey certain constraints, in order to facilitate tractable inference. This requires either to carefully design the structure by hand or to learn it from data [13, 20, 36, 46, 37, 53, 2, 14, 50, 42, 15, 33]. The special structural requirements of SPNs are opposed to the usual homogeneous structures employed in deep learning, and hinder a seamingless integration into deep learning frameworks. Additionally, learning SPN structures has proven hard to scale, precluding them from being used on e.g. large scale image tasks.

In this paper, *we investigate how important structure learning in SPNs actually is*. To this end, we introduce a simple and scalable method to construct *random and tensorized SPNs* (RAT-SPNs), waiving the necessity for structure learning: we first construct a random *region graph* [13, 36], which we subsequently populate with arrays of SPN nodes. This strategy essentially dictates a random hierarchical tensorial decomposition [48], leading to SPNs with reduced sparsity. RAT-SPNs map well onto deep learning frameworks like Tensorflow [1], scale to millions of parameters, and automatically taking advantage of GPU-parallelization.

For density estimation, i.e. the generative case, we use the classical expectation-maximization (EM) algorithm [12], which has recently been derived for SPNs [38]. Since EM is free of tuning-parameters and rapidly increases the likelihood, it is a natural choice for this task. We show that this simple strategy yields test-likelihoods surprisingly close to ID-SPN [46], one of the most sophisticated SPN learners available.

In addition, we show that RAT-SPNs, when trained discriminatively, yield classifiers competitive to deep neural nets. So far, no principled discriminative SPN structure learner is available while discriminative parameter learning has been mainly applied to images – relying either on powerful feature extraction [19] or specialized structures [3, 48, 43]. Our discriminative RAT-SPNs are domain-agnostic and thus applicable in a much wider setting.

Most importantly, *we demonstrate that RAT-SPNs deliver well-calibrated uncertainties*: they can be used to reliably detect anomalies and are robust under missing data. In contrast to deep classifiers, hybrid discrete-generative RAT-SPNs can explicitly quantify when they are not confident about their predictions. Furthermore, generative RAT-SPNs are not fooled by certain out-of-domain image detection tests on which VAEs, NFs, and ARDEs consistently fail [10, 34].

The start off by reviewing the required background and discussing related work. Subsequently, we introduce RAT-SPNs and our proposed tensorized learning schemes. Then, we thoroughly evaluate RAT-SPNs empirically w.r.t. current SPN learning approaches and deep neural nets for generative and discriminative modeling. Finally, we conclude and discuss future work.

2 BACKGROUND & RELATED WORK

We denote random variables (RVs) by upper-case letters, e.g. X, Y, and their values by corresponding lower-case letters, e.g., x, y. Similarly, we denote sets of RVs by upper-case bold letters, e.g., $\mathbf{X}$, $\mathbf{Y}$ and their combined values by corresponding lower-case letters, e.g., $\mathbf{x}$, $\mathbf{y}$.

An SPN $\mathcal{S}$ over $\mathbf{X}$ is a probabilistic model defined via a directed acyclic graph (DAG) containing three types of nodes: *input distributions*, *sums* and *products*. All leaves of the SPN are input distribution functions over some subset $\mathbf{Y} \subseteq \mathbf{X}$. Inner nodes are either weighted sums or products, denoted by S and P, respectively, i.e. $\mathsf{S} = \sum_{\mathsf{N} \in \mathbf{ch}(\mathsf{S})} w_{\mathsf{S},\mathsf{N}} \mathsf{N}$ and $\mathsf{P} = \prod_{\mathsf{N} \in \mathbf{ch}(\mathsf{P})} \mathsf{N}$, where $\mathbf{ch}(\cdot)$ denotes the children of a node. The sum weights $w_{\mathsf{S},\mathsf{N}}$ are assumed to be non-negative and normalized: $w_{\mathsf{S},\mathsf{N}} \geq 0$, $\sum_{\mathsf{N}} w_{\mathsf{S},\mathsf{N}} = 1$.

The *scope* of an input distribution N is defined as the set of RVs $\mathbf{Y}$ for which N is a distribution function, i.e. $\mathbf{sc}(\mathsf{N}) := \mathbf{Y}$. The scope of an inner (sum or product) node N is recursively defined as $\mathbf{sc}(\mathsf{N}) = \bigcup_{\mathsf{N}' \in \mathbf{ch}(\mathsf{N})} \mathbf{sc}(\mathsf{N}')$. To allow for efficient inference, SPNs should satisfy two structural constraints [11, 41], namely *completeness* and *decomposability*. An SPN is complete if for each sum S it holds that $\mathbf{sc}(\mathsf{N}') = \mathbf{sc}(\mathsf{N}'')$, for all $\mathsf{N}', \mathsf{N}'' \in \mathbf{ch}(\mathsf{S})$. An SPN is decomposable if it holds for each product P that $\mathbf{sc}(\mathsf{N}') \cap \mathbf{sc}(\mathsf{N}'') = \emptyset$, for all $\mathsf{N}' \neq \mathsf{N}'' \in \mathbf{ch}(\mathsf{P})$. In that way, all nodes in an SPN recursively define a distribution over their respective scopes: the leaves are distributions by definition, sum nodes are mixtures of their child distributions, and products are factorized distributions, assuming (conditional) independence among the scopes of their children.

Besides *representing* probability distributions, the crucial advantage of SPNs is that they permit *efficient inference*. For example, SPNs allow to compute arbitrary marginal distributions: In particular, let $\mathcal{S}(\mathbf{x})$ be a distribution over $\mathbf{X}$ represented by SPN $\mathcal{S}$, and let $\bar{\mathbf{X}} = \{X_{i_1}, \ldots, X_{i_M}\}$ be a set of RVs to be marginalized. The marginal distribution over $\mathbf{Z} = \mathbf{X} \setminus \bar{\mathbf{X}}$ can be computed as $\mathcal{S}(\mathbf{Z}) = \int_{x_{i_1}} \cdots \int_{x_{i_M}} \mathcal{S}(x_{i_1}, \ldots, x_{i_M}, \mathbf{Z}) \, \mathrm{d}x_{i_1} \ldots \mathrm{d}x_{i_M}$. As shown in [40], the integrals can be iteratively swapped with sums and distributed over products in the SPN, i.e. "pulled down" to the SPN leaves. Consequently, any marginalization task reduces to the corresponding marginalizations at the leaves (each leaf marginalizing only over its scope), and evaluating the internal nodes as usual in a bottom-up pass [40]. When the SPN uses only single-dimensional leaves, marginalization becomes particularly easy, by simply setting leaves corresponding to marginalized RVs to 1. Arbitrary *conditional distributions* can be computed in a similar manner. It is important to note that these inference scenarios are rendered tractable by the above mentioned structural constraints – completeness and decomposability – which are critical aspects when learning SPNs.

Indeed, SPN structure learning is a central topic in the literature, starting from [41], where an SPN structure tailored to images was proposed, based on recursive axis-aligned splits. Dennis and Ventura [13] improved this architecture by using non axis-aligned splits, using k-means applied to the transposed data matrix. Peharz et al. [36] introduced a bottom-up approach to learn SPN structures, using an information-bottleneck method. Gens and Domingos [20] proposed a general high-level scheme called *LearnSPN* which follows a hierarchical co-clustering approach, i.e. it alternately clusters data instances – corresponding to sum nodes – and splits variables – corresponding to product nodes – using independence tests. Since then, there have been several improvements of the basic LearnSPN scheme, such as regularization by employing multivariate leaves [53], employing an efficient SVD-approach [2], generating compacter networks by merging tree-structures into general DAGs [42], learning product nodes via multi-view clustering over variables [26] or lowering their complexity by approximate independence testing [16], and learning SPN structures over hybrid domains [33]. Rooshenas and Lowd [46] refined LearnSPN by learning leaf distributions using Markov networks represented by arithmetic circuits [32]. The resulting SPN learner, called ID-SPN, is state-of-the-art in density estimation on binary data, when considering single models (ensembles can improve results [30, 17]). In [48], a convolutional SPN tailored to image data was proposed, and Butz et al. [8] proposed a convolutional SPN variant interleaved with the structure proposed in [41].

While structure learning is indisputably a relevant topic in SPNs, the "antithesis" has received surprisingly little attention: *How important is detailed structure learning in SPNs really? Akin to deep neural networks, can we get decent models by just scaling up a random SPN structure and applying simple parameter estimation techinques?* The current success of deep learning makes this approach arguably worth exploring. Moreover, the special structural requirements of SPNs have probably hindered their wider use in practice, and in particular combinations with other deep learning models remain relatively unexplored. Random SPNs, as introduced in this paper, are therefore a promising direction for probabilistic deep learning.

3 RANDOM SUM-PRODUCT NETWORKS

In order to construct our *random and tensorized SPNs* (RAT-SPNs), we use the notion of a *region graph* [13, 36] as an abstract representation of the network structure. Given a set of RVs $\mathbf{X}$, a *region* $\mathbf{R}$ is defined as any non-empty subset of $\mathbf{X}$. Given any region $\mathbf{R}$, a *K-partition* $\mathcal{P}$ of $\mathbf{R}$ is a collection of K non-overlapping sub-regions $\mathbf{R}_1, \dots, \mathbf{R}_K$, whose union is again $\mathbf{R}$, i.e. $\mathcal{P} = \{\mathbf{R}_1, \dots, \mathbf{R}_K\}$, $\forall k\colon \mathbf{R}_k \neq \emptyset$, $\forall k \neq l\colon \mathbf{R}_k \cap \mathbf{R}_l = \emptyset$, $\bigcup_k \mathbf{R}_k = \mathbf{R}$. In this paper, we consider only 2-partitions, which causes all product nodes in our SPNs to have exactly two children. This assumption, frequently made in the SPN literature, simplifies SPN design and seems not to impair performance.

Algorithm 1 Random Region Graph

```
1: procedure RandomRegionGraph($\mathbf{X}, D, R$)
2:     Create an empty region graph $\mathcal{R}$
3:     Insert $\mathbf{X}$ in $\mathcal{R}$
4:     for $r = 1 \dots R$ do
5:         Split($\mathcal{R}, \mathbf{X}, D$)

1: procedure Split($\mathcal{R}, \mathbf{R}, D$)
2:     Draw balanced partition $\mathcal{P} = \{\mathbf{R}_1, \mathbf{R}_2\}$ of $\mathbf{R}$
3:     Insert $\mathbf{R}_1, \mathbf{R}_2$ in $\mathcal{R}$
4:     Insert $\mathcal{P}$ in $\mathcal{R}$
5:     if $D > 1$ then
6:         if $|\mathbf{R}_1| > 1$ then Split($\mathcal{R}, \mathbf{R}_1, D - 1$)
7:         if $|\mathbf{R}_2| > 1$ then Split($\mathcal{R}, \mathbf{R}_2, D - 1$)
```

A *region graph* $\mathcal{R}$ over $\mathbf{X}$ is a connected DAG whose nodes are regions and partitions such that i) there is exactly one region $\mathbf{R} = \mathbf{X}$ without parents (i.e. $\mathbf{X}$ is the *root region*), ii) all leaves of $\mathcal{R}$ are regions, iii) all children of regions are partitions and all children of partitions are regions (i.e. $\mathcal{R}$ is bipartite), iv) if $\mathcal{P}$ is a child of $\mathbf{R}$, then $\bigcup_{\mathbf{R}' \in \mathcal{P}} \mathbf{R}' = \mathbf{R}$ and v) if $\mathbf{R}$ is a child of $\mathcal{P}$, then $\mathbf{R} \in \mathcal{P}$.

Given a region graph, we can easily construct a corresponding SPN as follows: Populate each leaf-region with a collection of I input distributions, and all other regions with a collection of sum nodes. For the root region we create C sum nodes, and for all internal regions, we create S sum nodes. Finally, for all partitions, take all cross-products of nodes contained in the child-regions, and connect these products as children of all sums in the parent region. Pseudo-code for this procedure is provided in the supplementary.

We denote the C sum nodes in the root region as $\mathcal{S}_c(\mathbf{X})$, $c = 1, \dots, C$. For density estimation, we assume $C = 1$, in which case the single root readily represents a correctly normalized density $\mathcal{S}(\mathbf{X}) := \mathcal{S}_1(\mathbf{X})$. For classification, the $C > 1$ roots represent *class-conditional* distributions $\mathcal{S}_c(\mathbf{X}) =: \mathcal{S}(\mathbf{X} \,|\, Y = y)$, $y \in \{1, \dots, C\}$. A sample $\mathbf{x}$ is classified by applying Bayes' rule: $\mathcal{S}(Y \,|\, \mathbf{x}) = \frac{\mathcal{S}(\mathbf{x} \,|\, Y)\, P(Y)}{\mathcal{S}(\mathbf{x})} = \frac{\mathcal{S}(\mathbf{x} \,|\, Y)\, P(Y)}{\sum_y \mathcal{S}(\mathbf{x} \,|\, y)\, P(y)}$. The class-prior $P(Y)$ can be estimated from the empirical class-distribution, or just be fixed to, e.g., uniform. The *marginal data-likelihood* $\mathcal{S}(\mathbf{x}) = \sum_y \mathcal{S}(\mathbf{x} \,|\, y)\, P(y)$ is also a useful quantity, as it allows us to detect outliers: In the case that a classifier is fed with a sample which is far from any training data, we can expect $\mathcal{S}(\mathbf{x})$ to be low.

We construct random regions graphs – and thus RAT-SPNs – with the simple procedure depicted in Algorithm 1: We randomly divide the root region into two sub-regions of equal size (possibly breaking ties) and proceed recursively until depth D, resulting in an SPN of depth $2D$. This recursive splitting mechanism is repeated R times. An example of a RAT-SPN is illustrated in the supplementary.

It is easy to verify that the number of sum-weights in RAT-SPNs is given as $W_{\mathsf{S}} =$

$$\begin{cases} RCI^2 & \text{if } D = 1, \\ R\left(CS^2 + (2^{D-1} - 2)S^3 + 2^{D-1}SI^2\right) & \text{if } D > 1. \end{cases} \quad (1)$$

Similarly, we can count the parameters of the input distributions, which we assume to factorize into univariate distributions. In this case, it follows that the total number of parameters for the input distributions is

$$W_{\mathsf{D}} = RI|\mathbf{X}|P, \quad (2)$$

where P is the number of parameters per univariate distribution.

We implemented Alg. 1 in Python and the corresponding RAT-SPNs in Tensorflow.[1] The input distributions are Gaussians for real data and categorical for discrete data. All computation are performed in the log-domain to avoid numerical underflow. Sum-weights, which are required to be non-negative and normalized, are re-parameterized via log-softmax layers. To perform summations in the log-domain, we use the *log-sum-exp* trick. In this paper, we consider both *generative* and *discriminative* learning, as discussed in the following.

3.1 GENERATIVE LEARNING

For generative learning, we assume that we have a training set $\mathcal{X} = \{\mathbf{x}_1, \dots, \mathbf{x}_N\}$ of i.i.d. samples drawn from an unknown distribution $P^*(\mathbf{X})$, which we wish to approximate. The canonical approach to generative learning is maximizing the log-likelihood

$$\mathsf{LL}(\boldsymbol{w}) = \frac{1}{N} \sum_{n=1}^{N} \log \mathcal{S}(\mathbf{x}_n), \quad (3)$$

where $\boldsymbol{w}$ denotes all parameters of the SPN, i.e. sum-weights and parameters of the input distributions. Note that by construction, $\mathcal{S}(\mathbf{X})$ is already a correctly normalized distribution over $\mathbf{X}$.

To optimize (3), we use the standard Expectation-Maximization (EM) algorithm [12], which has been recently derived for SPNs in [38]. EM rapidly and monotonically increases the likelihood, is free of tuning-parameters and can be implemented via simple forward and backward evaluations to compute the required expected sufficient statistics – see [38] for details. Due to these convenient properties, we use EM for the generative case. Note that the concave-convex procedure proposed in [54] coincides with EM updates for sum-weights, but is in general distinct for input distributions.

[1] `https://github.com/cambridge-mlg/RAT-SPN`

3.2 DISCRIMINATIVE LEARNING

For discriminative learning, we focus on classification. Let $\mathcal{X} = \{(\mathbf{x}_1, y_1), \dots, (\mathbf{x}_N, y_N)\}$ be a training set of inputs $\mathbf{x}_n$ and class labels y_n. We train RAT-SPN classifiers by minimizing the *cross-entropy*

$$\mathsf{CE}(\boldsymbol{w}) = -\frac{1}{N} \sum_{n=1}^{N} \log \frac{\mathcal{S}_{y_n}(\mathbf{x}_n)}{\sum_{y'} \mathcal{S}_{y'}(\mathbf{x}_n)}, \quad (4)$$

which is equivalent to maximizing the *conditional log-likelihood* $\sum_n \log \mathcal{S}(y_n \,|\, \mathbf{x}_n)$, when assuming a uniform class prior. Furthermore, we can readily combine (3) and (4) into a *hybrid generative-discriminative* [4] loss

$$\mathsf{H}(\boldsymbol{w}) = \lambda\, \mathsf{CE}(\boldsymbol{w}) - (1 - \lambda)\, \frac{\mathsf{LL}(\boldsymbol{w})}{|\mathbf{X}|}, \quad (5)$$

which trades off cross-entropy and log-likelihood. For $\lambda = 1$, we retrieve pure discriminative learning, while for $\lambda = 0$, we retrieve pure generative learning. For $0 < \lambda < 1$, we are allowing our RAT-SPN classifiers to also capture the distribution over $\mathbf{X}$, a crucial feature to deal with uncertainty over inputs, e.g., in presence of missing values. The likelihood LL is obtained by marginalizing the class variable Y, as illustrated above. For discriminative learning, we use Adam with default hyper-parameters and a fixed batchsize of 100.

3.3 PROBABILISTIC DROPOUT

The size of RAT-SPNs can be easily controlled via the structural parameters D, R, S and I. As usual in deep learning, we design RAT-SPNs structures to be overparameterized. In order to prevent overfitting, we perform *early stopping* by monitoring the loss on a validation set. We monitor the objective of interest on a validation set, i.e. the log-likelihood for the generative case or the classification rate for the discriminative case, and save the current model whenever we get an improvement over the previous best model. Furthermore, we propose two variants of the *dropout* heuristic [49] for RAT-SPNs: at inputs and at sum nodes.

Dropout at inputs essentially marks input features as missing at random. Following the probabilistic paradigm, we simply wish to marginalize over these missing features. Fortunately, this is an easy task in

SPNs, as we only need to set the input distributions corresponding to a dropped-out features to value 1. A similar criterion was used in a convolutional variant of SPNs [48], which drops out small image patches, however.

We introduce *dropout at sum nodes*, by setting their child-products to 0 (in fact $-\infty$ in log-domain) with a certain probability. This effectively introduces artificial information to the latent variables associated to the mixtures represented by sum nodes [38] by setting the probability of a random subset of states to 0.

4 EXPERIMENTS

We evaluated RAT-SPNs on a wide range of tasks and real world benchmarks. First, we investigated their capability as density estimators in the generative setting, comparing them to state-of-the-art SPN learners, VAEs and Masked Autoencoders (MADEs) [21]. Second, we compared RAT-SPNs with deep neural networks in the discriminative setting, over a diverse set of classification domains. Moreover, we analysed the uncertainties represented by RAT-SPNs, as employed for anomaly detection and classification under missing inputs, two scenarios on which current deep architectures fall short [34, 10].

4.1 GENERATIVE LEARNING: RAT-SPNs ARE COMPARABLE TO STATE-OF-THE-ART

For the generative setting, we evaluated RAT-SPNs on 20 benchmark datasets, commonly used to compare SPN learners [20]. The main objective in this experiment is not necessarily to yield new state-of-the-art log-likelihoods on these datasets. Rather, we aim to investigate to which extent sophisticated SPN learning schemes are actually able to significantly improve over our simple approach, using random over-parametrized SPN.

To this end, we compared RAT-SPNs; LearnSPN [20], the most prominent SPN structure learner; LearnSPN-RGVS [16], an extension to LearnSPN, which approximates the statistical tests for product nodes in a random fashion; and OBMM [44], using LearnSPN-like randomly generated structures and Bayesian parameter learning.[2] Consequently, we compared RAT-SPNs against full structure learning (LearnSPN), a randomly-flavored variant (LearnSPN-RGVS), and random structure with sophisticated parameter learning (RandSPN+OBMM). Additionally, we report state-of-the-art log-likelihood as achieved by ID-SPN [46] for structure learning, MADEs with 8 variable orderings [21] and VAEs with 5 importance weighted samples [6]. IDSPN additionally uses SPN leaves with direct variable interactions, and MADEs and VAEs are more flexible density representations which, however, facilitate only sampling and evaluating (a lower bound of) the density.

[2]OBMM is the only other approach, which also employs random structures. However, it does not compile to computation graphs and does not make use of deep neural learning techniques.

Table 1: Average test log-likelihoods on 20 datasets. Best results for each dataset are in bold (within SPN learners using single-dimensional leaves). Within the group LearnSPN/RAT-SPN/ID-SPN, results which are not significantly worse than the best, are marked with ∘.

	LearnSPN	RGVS	OBMM	RAT-SPN	ID-SPN	MADE	VAE
nltcs	-6.11	-6.37	-6.07	∘**-6.01**	∘-6.02	-6.04	-5.99
msnbc	-6.11	-6.11	**-6.03**	∘-6.04	∘-6.04	-6.06	-6.09
kdd-2k	∘-2.18	–	-2.14	∘**-2.13**	∘-2.13	-2.07	-2.12
plants	**-12.99**	-16.78	-15.14	-13.44	-12.54	-12.32	-12.34
jester	-53.48	-54.97	-53.86	∘**-52.97**	∘-52.86	-52.23	-51.54
audio	∘-40.50	-41.94	-40.70	∘**-39.96**	∘-39.79	-38.95	-38.67
netflix	-57.328	-59.84	-57.99	**-56.85**	-56.36	-55.16	-54.73
accid.	**-30.04**	-40.23	-42.66	-35.49	-26.98	-26.42	-29.11
retail	∘-11.04	-11.34	-11.42	∘**-10.91**	∘-10.85	-10.81	-10.83
pumsb.	**-24.78**	-42.42	-45.27	-32.53	-22.41	-22.3	-25.16
dna	**-82.52**	-99.27	-99.61	-97.23	-81.21	-82.77	-94.56
kosarek	∘-10.99	-11.49	-11.22	∘**-10.89**	∘-10.6	–	-10.64
msweb	-10.25	-11.00	-11.33	**-10.12**	-9.73	-9.59	-9.727
book	∘-35.89	-35.67	-35.55	∘**-34.68**	∘-34.14	-33.95	-33.19
e.movie	∘**-52.49**	-64.46	-59.50	-53.63	∘-51.51	-48.7	-47.43
web-kb	∘-158.204	-167.55	-165.57	∘**-157.53**	∘-151.84	-149.59	-146.9
reut.52	∘**-85.07**	-97.27	-108.01	∘-87.37	∘-83.35	-82.80	-81.33
20ng	-155.93	–	-158.01	**-152.06**	-151.47	-153.18	-146.9
bbc	∘**-250.69**	-269.03	-275.43	∘-252.14	∘-248.93	-242.40	-240.94
ad	∘**-19.73**	-57.55	-63.81	-48.47	∘-19.05	-13.65	-18.81

We cross-validated the split-depth $D \in \{1, 2, 3, 4\}$ and the number of sum-weights $W_S \in \{10^3, 10^4, 10^5\}$. In order to yield a particular W_S, we used (1) to select appropriate values for R, S and I. These values were picked a-priori such that they were roughly balanced and approximately yielded a targeted W_S (see supplementary), but *not* tuned to the validation set. We used soft EM for 100 epochs and used early stopping for regularization. No dropout was applied in the generative case.

Average test log-likelihoods are presented in Tab. 1. The largest log-likelihood among direct competitors is in bold for each dataset. We furthermore tested for statistical significance within the group RAT-SPN, LearnSPN, and ID-SPN[3] where we denote with ∘ results which are not significantly worse than the best one (according to a two-sample t-test, $p = 0.05$).

The results in Tab. 1 are surprising, as the log-likelihoods of RAT-SPN are often close to the ones of ID-SPN. In fact, ID-SPN is significantly better than RAT-SPN on only 7 out of 20 datasets. Moreover, RAT-SPNs are only on 5 datasets more than 5% worse, relative to ID-SPN. Given that RAT-SPNs do not use *any* structure learning at all, while ID-SPN is a highly sophisticated struc-

[3]For RGVS, CCCP, and OBMM, we unfortunately had no sample-wise results, so no significance test could be conducted.

dataset	domain	C	#feat.	#train	#val.	#test
mnist	image	10	784	54k	6k	10k
f-mnist	image	10	784	54k	6k	10k
imdb	text	2	200	20k	5k	25k
theorem	logic	6	51	3670	1224	1224
20ng	text	20	50	13568	1508	3770
higgs	physics	2	28	9M	1M	1M
wine	chem.	2	11	3899	1299	1299

Table 2: Overview of classification datasets.

dataset	GMM	RAT-SPN	MLPd	MLP+
mnist	97.37	○**98.29**	98.05	○98.52
f-mnist	88.08	89.43	**89.89**	90.63
imdb	○75.65	○**75.90**	○75.72	○75.83
theorem	○55.64	○55.47	○**57.76**	○56.21
20ng	47.61	○**48.49**	○**48.49**	○48.97
higgs	74.14	73.82	**76.36**	76.45
wine	○77.21	○77.14	○**77.83**	○79.45

Table 3: Test classification accuracy, best values among GMM, RAT-SPN, and MLPd in bold. Results which are not significantly different (according to McNemar's test) from the best are denoted by ○.

ture learner, the difference is indeed surprisingly small. On three datasets RAT-SPNs even perform better than ID-SPN, although not significantly. Moreover, RAT-SPNs almost consistently outperform OBMM, except on 'msnbc'. On 8 datasets, OBMM performs more than 5% worse, relative to RAT-SPNs. Given that OBMM is the only other approach using random structures, we find that RAT-SPNs establish state-of-the-art for SPNs with random structures. One should note, that this comparison to OBMM is not entirely fair, since RAT-SPNs explore much larger structures, and are also not restricted to trees. However, our hypothesis for this paper was that overparameterized SPNs with simple parameter learning deliver satisfying results. We find that the results in Table 1 confirm this hypothesis.

4.2 DISCRIMINATIVE LEARNING: RAT-SPNs ARE COMPETITIVE WITH NEURAL NETS

Next, we evaluated the discriminative performance of RAT-SPNs. This time, the natural competitors are deep neural networks, as discriminative structure learning for SPNs has been largely unexplored so far. To this end, we apply RAT-SPNs to 7 classification tasks from various domains. Tab. 2 summarizes the characteristics of these datasets. See supplementary for additional details.

Due to their random nature, RAT-SPNs are *domain agnostic*, i.e. they do not have an inductive bias tailored towards any particular type of data, as opposed to e.g. convolutional neural networks for images. Clearly, incorporating convolutional structures in SPNs would be advantageous for 'mnist' and 'fashion-mnist', as demonstrated in [48, 8]. However, the model-agnostic character of RAT-SPNs allows their use in a wider range of problems, and in particular their performance would not degrade if the pixels of '(fashion-)mnist' were scrambled. As input distributions we used Gaussians with variance fixed to 1.

We compared RAT-SPNs to multi-layer perceptrons (MLPs) with rectified linear units, trained MLPs in two variants, namely a standard variant using only dropout (MLPd) – like in RAT-SPNs – and a variant (MLP+) also employing Xavier-initialization [22] and batch normalization [25]. The latter includes two additional training techniques, which have evolved over decades, while similar techniques for RAT-SPNs are not yet available. Thus, MLPd might serve as a fairer comparison to RAT-SPNs.

For both RAT-SPNs and MLPs, we cross-validated the "depth" (number of hidden layers for MLPs, and split-depth D for RAT-SPNs), and the "width" (number of hidden units for MLPs, and parameters R, S and I for RAT-SPNs). Thereby, we first selected suitable ranges for the MLP's hyper-parameters and *then* matched the sizes of the RAT-SPN. Thus, the comparison is fair in terms of considered depth and number of model parameters. The complete hyperparameter configurations are reported in the supplementary.

All models were trained for 200 epochs, optimizing cross-entropy using Adam in its default setting and a batchsize of 100. For regularization, we applied early stopping and dropout-rates $\{0.25, 0.5, 0.75, 1.0\}$, independently for inputs and hidden layers/sum layers. For 'higgs', we only trained one epoch due to the large number of samples, i.e. we effectively considered an online setting. We further compared to Gaussian mixture models (GMMs) with a massive number of components, namely 1000, 2000, 4000, and 8000. In this way, GMMs provide a "shallow" classification baseline for SPNs. The number of components was cross-validated as well as the dropout-rates at the inputs – dropout was applied in similar fashion as for RAT-SPNs. For the covariance matrices, we used the unity matrix.

Tab. 3 summarizes the classification performances on the test sets. We see that RAT-SPNs compare well to MLPd. Out of the 7 dataset, RAT-SPNs win 2 times against MLPd and have one draw (the number of correct examples for 20ng was indeed exactly the same). Moreover, RAT-SPNs are only twice significantly worse than MLP+. We see that GMMs tend to perform slightly better than RAT-SPNs on datasets with few variables. On

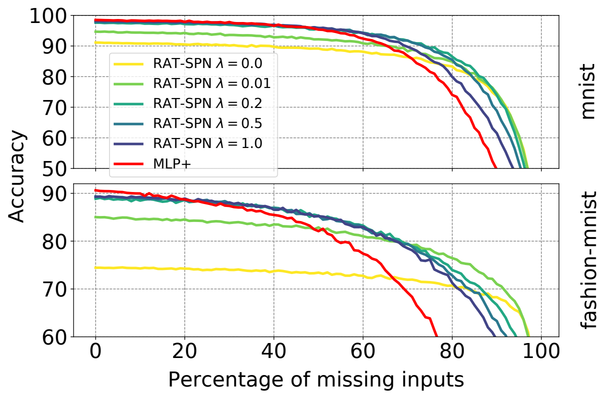

Figure 1: Classification accuracy of hybrid RAT-SPNs and MLP+ over percentage p of missing inputs, on mnist (top) and fashion-mnist (bottom). For better readability, only the accuracy range 50%-100% (resp. 60%-100%) is shown for mnist (resp. fashion-mnist).

the datasets with many variables, however, RAT-SPNs perform considerably better. This is consistent with the well-known fact that GMMs do not scale well to high-dimensional spaces. Overall, we see that RAT-SPNs deliver decent classifiers when trained discriminatively. So far, most works on discriminative parameter estimation for SPNs were tailored to images, exploiting either powerful pre-extracted features [19] or using specialized structures [48, 3]. Our results are the first, which investigate the effectiveness of SPNs when trained end-to-end using entirely random structures. We do not only scale SPN training to the regime of deep neural learning, but also demonstrate it to be competitive with deep networks.

However, as shown next, RAT-SPNs have several advantages over deep neural networks, due to the fact that they represent a *tractable* full joint distribution over *both* inputs $\mathbf{X}$ and class Y. Since a purely discriminative model, i.e. optimized only for cross-entropy, is not RAT-SPNs are not encouraged to capture the distribution over inputs $\mathbf{X}$ well, we performed hybrid generative-discriminative post-training on our RAT-SPN classifiers. Specifically, we applied Adam for 20 additional epochs, optimizing the hybrid objective (5) for various setting of $0 \leq \lambda \leq 1$. For λ close to 0, we get higher test-likelihoods and lower classification accuracies (generative flavor) than for λ close to 1 (discriminative flavor). This trade-off, illustrated in the supplementary, is consistent with literature on hybrid generative-discriminative learning [39, 47].

4.3 RAT-SPNs ARE ROBUST UNDER MISSING FEATURES

When input features in $\mathbf{X}$ are missing at random, we ideally want to marginalize them [31]. As SPNs allow efficient marginalization, they should be robust under missing features, especially for smaller λ (more generative character). To this end, we discard pixels with probability p in the test samples for mnist and fashion mnist and classify them using RAT-SPNs. Note that marginalizing missing features amounts to probabilistic dropout used during training, i.e. simply setting corresponding input distributions to 1. Similarly, we might expect MLPs to perform robustly under missing features, by applying (classical) dropout during test time. Alternatively, missing data can be treated with e.g. k-nearest neighbor imputation. This, however, requires one to store the whole training set and to solve an expensive nearest neighbor search for each test sample.

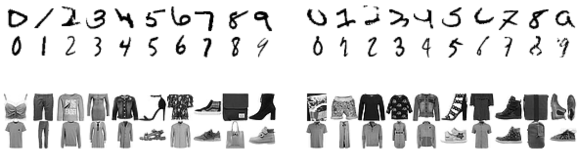

Figure 2: Examples of outliers (respective top row) and inliers (respective bottom row) for 'mnist' and 'fashion-mnist', for each class. Samples in the left column were classified correctly, while samples in the right columns were classified incorrectly.

Fig. 1 summarizes the classification results when varying the fraction of missing features p between 0.0 and 0.99. As one can see, RAT-SPNs are more robust than MLP+ using dropout. This effect becomes stronger with smaller λ, i.e. for models with a "more generative flavor". A particularly interesting choice is $\lambda = 0.2$: here the corresponding RAT-SPN starts with an accuracy of 97.61% for no missing features and degrades very gracefully: for a large fraction of missing features ($> 60\%$) the advantage over MLP+ is dramatic.

4.4 RAT-SPNs KNOW WHAT THEY DON'T KNOW

Besides being robust under missing features, an important feature of (hybrid) generative models is that they are naturally able to detect outliers and peculiarities by monitoring the marginal likelihood over inputs $\mathbf{X}$. Our aim in this section is to demonstrate that RAT-SPNs readily provide well-calibrated uncertainties for this purpose. We first focus on classification, where the marginal likelihood of RAT-SPNs offer a principled outlier signal, unlike deep neural classifiers. Second, we investigate the ability of RAT-SPNs to perform anomaly detection on certain image datasets, which have recently been identified as being problematic cases for deep generative models [34, 10].

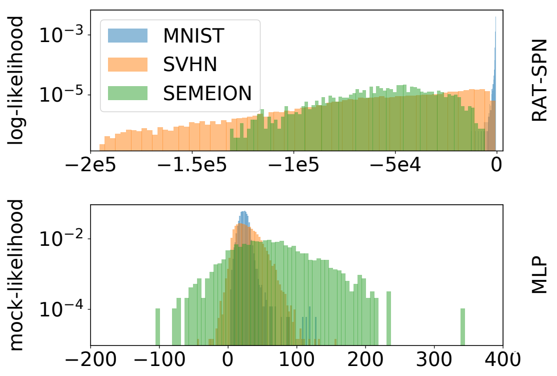

Figure 3: Histograms of test log-likelihoods for 'mnist', 'svhn' and 'semeion' data for RAT-SPN (top) and corresponding computations performed for MLP+ ("mock-likelihood") (bottom). Both models were trained on 'mnist'. The likelihoods of RAT-SPNs yield a strong signal whether a sample is in-domain or out-of-domain.

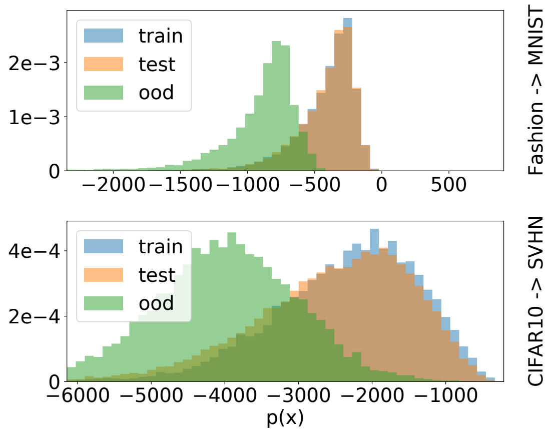

Figure 4: Histograms the log-likelihoods of RAT-SPNs on the native data set (blue: train, orange: test) and out-of-domain (ood) data set (green). Top: Native data 'fashion-mnist', out-of-domain data 'mnist'. Bottom: Native data 'cifar10', out-of-domain data 'svhn'. Cf. Fig. 2 in [34] for results for VAEs and GLOW.

For the classification setting, we evaluated the marginal likelihoods on the test set for both 'mnist' and 'fashion-mnist', using the respective RAT-SPN post-trained with $\lambda = 0.2$. For illustrative purposes, we divided the test samples into correctly and incorrectly classified ones. From both groups, we selected two examples for each class, namely the one with the lowest input probability (outlier) and the one with the highest input probability (inlier). This yielded 4 groups of 10 samples each: outlier/correct, outlier/incorrect, inlier/correct, inlier/incorrect. These samples are shown in Fig. 2 (a higher resolution version is provided in the supplementary).

Albeit qualitative, these results are interesting: For 'mnist', one can visually confirm that the outlier digits are indeed peculiar, both the correctly and the incorrectly classified ones. For instance, in the outlier/incorrect group the '0' and '3' have been apparently cut off during pre-processing, and the '6' is not recognizable for humans. In the inlier/incorrect group we have rather ambiguous examples, which seems to be the major cause of misclassification. This is reflected by the fact that the predictive uncertainty (cross-entropy of the class distribution) was highest in this group, and that in 8 out of 10 cases the correct class had the second highest probability (see supplementary). Similar results hold for 'fashion-mnist'.

For a more quantitative analysis, we used a variant of *transfer testing* proposed by Bradshaw *et al.* [5]. This technique is quite simple: we feed a classifier trained on one domain (e.g. 'mnist') with examples from a related but different domain, e.g. street view house numbers ('svhn') [35] or the handwritten digits of 'semeion' [7], converted to 'mnist' format (28×28 pixels, grey scale). While we would expect that most classifiers perform poorly in such setting, an AI system should be aware that it is confronted with out-of-domain data. While Bradshaw *et al.* applied this technique to output uncertainties, it is clearly also applicable to *input uncertainties*, i.e. the marginal probability of features $\mathbf{X}$.

Fig. 3(top) shows histograms of the log-probabilities over inputs for the RAT-SPN post-trained with $\lambda = 0.2$, when fed with 'mnist' test data (in-domain), 'svhn' test data (out-of-domain) and 'semeion' (out-of-domain). The likelihood histograms provide a strong signal whether a sample comes from in-domain or out-of-domain. In fact, the samples from 'mnist' and 'semeion' can be perfectly discriminated, i.e. the highest input probability in 'semeion' is smaller than the lowest input probability in 'mnist'. The samples of 'mnist' and 'svhn' overlap by less than 1%. Consequently, RAT-SPNs – and other tractable joint probability models – have an extra communication channel to inform us whether we ought to trust their predictions.

However, a potential caveat might be: Does this result indeed stem from the fact that we model a full joint distribution, or merely from the fact that we average outputs of a classifier?[4] Thus, as a sanity check, we performed the likewise computations in the trained MLP+. One might suspect that the result, although not interpretable as log-probability, still yields a decent signal to detect outliers. In need of a name for this exotic quantity, we name it

[4] Assuming uniform class prior, marginalizing the class variable from the RAT-SPN corresponds to averaging its outputs.

mock-likelihood. Fig. 3(bottom) shows histograms of this mock-likelihood: although more spread, histograms for out-of-domain data are highly overlapping and do not yield a clear signal for out-of-domain vs. in-domain.

We apply a similar line of reasoning for outlier detection in the generative case, and investigated if RAT-SPNs are susceptible to the "likelihood mirage" affecting several deep generative models such as VAEs, ARDEs and NFs: In [10, 34], it has recently been noted that samples from certain test image datasets are not only hard to be recognized as out-of-domain, but are consistently deemed to be even more likely than in-domain samples. In [34] this effect has been reported for VAEs, PixelCNNs [52], GLOW [27] for image data that is clearly – at least for humans – very different from the target test. This behavior is quite unexpected, since VAEs, PixelCNNs and GLOW – in contrast to MLP classifiers – are generative models and trained to maximise the likelihood over features $\mathbf{X}$. Note that likelihood has classically been considered a proper score for anomaly detection [9, 23].

We replicate the experimental setting of [34] by training a generative RAT-SPN on the training sets of 'fashion-mnist' and 'cifar10'. We then evaluate the likelihood of in-domain test samples (belonging to the same dataset) and of out-of-domain samples coming from 'mnist' and 'svhn', respectively. Fig. 4 reports the histogram of the log-likelihoods RAT-SPNs used to score train and test in-domain and out-of-domain samples for 'fashion-mnist' $\rightarrow$ 'mnist' (top) and 'cifar10' $\rightarrow$ 'svhn' (bottom).

Differently from VAE, PixelCNN and GLOW (cf. [34] for corresponding plots), RAT-SPNs are not assigning higher likelihoods to out-of-domain samples and clearly discriminate among inliers and outliers. This is evident for 'mnist' against 'fashion-mnist' and slightly less prominent in the other case where 'svhn' likelihood histogram overlaps slightly more with 'cifar10' ones. In any case, this clearly highlights the ability of RAT-SPNs to properly calibrate uncertainties when compared to current deep generative models based on neural networks, which fall prey to the "likelihood mirage".

5 CONCLUSION

We have proposed a simple approach to employ SPNs for deep learning, and demonstrated that tractable models like SPNs can get surprisingly far even without sophisticated structure learning. Specifically, our simple and scalable approach to construct a random but valid SPN structure, tensorize it, and combine it with simple training mechanisms like soft EM or Adam delivers results comparable to state-of-the-art, both in the generative and the discriminative setting. This represents a *tremendous simplification* of learning SPNs and in turn paves the way to a wider use of tractable probabilistic models in the deep learning community.

By implementing RAT-SPNs in Tensorflow, we automatically make use of GPU computations leading to considerable speed-ups compared to traditional SPN learning on CPUs. For example, one epoch on 'mnist' takes roughly a minute for a RAT-SPN with depth 2 and 1.2M parameters, using a GTX 1080Ti. This is a speedup of 45 X compared to a single CPU. However, a comparable MLP with 1.2M parameters only needs slightly more than 1 sec/epoch. This is not surprising, as MLPs rely on highly parallelized matrix multiplications and efficient non-linearities. On the other hand, RAT-SPNs bring enhanced sparsity in the weight matrix to establish consistency across any marginals and, therefore, make the computation less efficient on GPUs. Moreover, they employ expensive log-sum-exp computations, used to avoid numerical underflow. To speed-up RAT-SPNs, one can approximate them in each region with a sparsified variant. This avoids to generate all cross-products reducing the number of operands involved. One could also perform operations in the linear domain, together with a smart rescaling approach to avoid numerical underflow. Furthermore, we are currently investigating approaches using specialized hardware such as FPGAs for SPNs.

Overall, the ideas and results presented in this paper are promising directions for probabilistic deep learning. As demonstrated, SPNs are capable connectionist models with additional advantages like calibrated anomaly detection, treatment of missing features, or most importantly, the power of tractable probabilistic inference. Exploring these feature jointly with deep neural networks, e.g. as calibrated loss layers, is the perhaps the most promising avenue for future work.

Acknowledgements

RP acknowledges that 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. KK acknowledges the support of the Rhine-Main Universities Network for "Deep Continuous-Discrete Machine Learning" (DeCoDeML). KK and XS acknowledge the funding due to the Deutsche Forschungsgemeinschaft (DFG) project "CAML", Grant Agreement No. 1686/3-1.

References

[1] Abadi, M. et al. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015.

[2] T. Adel, D. Balduzzi, and A. Ghodsi. Learning the structure of sum-product networks via an SVD-based algorithm. In *Proceedings of UAI*, pages 32–41, 2015.

[3] M. R. Amer and S. Todorovic. Sum product networks for activity recognition. *IEEE transactions on pattern analysis and machine intelligence*, 38(4):800–813, 2016.

[4] G. Bouchard and B. Triggs. The Trade-Off between Generative and Discriminative Classifiers. In *COMPSTAT*, pages 721–728, 2004.

[5] J. Bradshaw, A. Matthews, and Z. Ghahramani. Adversarial examples, uncertainty, and transfer testing robustness in Gaussian process hybrid deep networks. *preprint arXiv*, 2017. arxiv.org/abs/1707.02476.

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

[7] M. Buscema. *MetaNet*: The Theory of Independent Judges*, volume 33. Taylor & Francis, 1998.

[8] C. J. Butz, J. S. Oliveira, A. E. dos Santos, and A. L. Teixeira. Deep convolutional sum-product networks. In *Proceedings of AAAI*, 2019.

[9] V. Chandola, A. Banerjee, and V. Kumar. Anomaly detection: A survey. *ACM computing surveys (CSUR)*, 41(3):15, 2009.

[10] H. Choi and E. Jang. Generative ensembles for robust anomaly detection. *arXiv preprint arXiv:1810.01392*, 2018.

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

[12] A. Dempster, N. Laird, and D. Rubin. Maximum likelihood from incomplete data via the EM algorithm. *J. Royal Statistical Society, Series B*, 39(1):1–38, 1977.

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

[14] A. Dennis and D. Ventura. Greedy structure search for sum-product networks. In *IJCAI*, pages 932–938, 2015.

[15] A. Dennis and D. Ventura. Online structure-search for sum-product networks. In *Proceedings of ICML*, pages 155–160, 2017.

[16] N. Di Mauro, F. Esposito, F. G. Ventola, and A. Vergari. Sum-product network structure learning by efficient product nodes discovery. *Intelligenza Artificiale*, 12(2):143–159, 2018.

[17] N. Di Mauro, A. Vergari, T. Basile, and F. Esposito. Fast and accurate density estimation with extremely randomized cutset networks. In *Proceedings of ECML/PKDD*, pages 203–219, 2017.

[18] L. Dinh, J. Sohl-Dickstein, and S. Bengio. Density estimation using real NVP. *arXiv preprint arXiv:1605.08803*, 2016.

[19] R. Gens and P. Domingos. Discriminative learning of sum-product networks. In *Proceedings of NIPS*, pages 3248–3256, 2012.

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

[21] M. Germain, K. Gregor, I. Murray, and H. Larochelle. MADE: Masked autoencoder for distribution estimation. In *Proceedings of ICML*, pages 881–889, 2015.

[22] X. Glorot and Y. Bengio. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of AISTATS*, pages 249–256, 2010.

[23] M. Goldstein and S. Uchida. A comparative evaluation of unsupervised anomaly detection algorithms for multivariate data. *PloS one*, 11(4):e0152173, 2016.

[24] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of NIPS*, pages 2672–2680, 2014.

[25] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Prooceedings of ICML*, pages 448–456, 2015.

[26] P. Jaini, A. Ghose, and P. Poupart. Prometheus: Directly learning acyclic directed graph structures for sum-product networks. In *Proceedings of PGM*, 2018.

[27] D. P. Kingma and P. Dhariwal. Glow: Generative flow with invertible 1x1 convolutions. In *Proceedings of NeuRIPS*, pages 10236–10245, 2018.

[28] D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *ICLR, https://arxiv.org/abs/1312.6114*, 2014.

[29] H. Larochelle and I. Murray. The neural autoregressive distribution estimator. In *Proceedings of AISTATS*, pages 29–37, 2011.

[30] Y. Liang, J. Bekker, and G. Van den Broeck. Learning the structure of probabilistic sentential decision diagrams. In *Proceedings of UAI*, 2017.

[31] R. J. A. Little and D. B. Rubin. *Statistical analysis with missing data*, volume 333. John Wiley & Sons, 2014.

[32] D. Lowd and A. Rooshenas. Learning Markov networks with arithmetic circuits. *Proceedings of AISTATS*, pages 406–414, 2013.

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

[34] E. Nalisnick, A. Matsukawa, Y. W. Teh, D. Gorur, and B. Lakshminarayanan. Do deep generative models know what they don't know? *ICLR, https://arxiv.org/abs/1810.09136*, 2019.

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

[36] R. Peharz, B. Geiger, and F. Pernkopf. Greedy part-wise learning of sum-product networks. In *Proceedings of ECML/PKDD*, volume 8189, pages 612–627, 2013.

[37] R. Peharz, R. Gens, and P. Domingos. Learning selective sum-product networks. In *ICML-LTPM Workshop*, 2014. online: https://sites.google.com/site/ltpm2014/.

[38] R. Peharz, R. Gens, F. Pernkopf, and P. Domingos. On the latent variable interpretation in sum-product networks. *IEEE TPAMI*, 39(10):2030–2044, 2017.

[39] R. Peharz, S. Tschiatschek, and F. Pernkopf. The most generative maximum margin Bayesian networks. In *Proceedings of ICML*, pages 235–243, 2013.

[40] R. Peharz, S. Tschiatschek, F. Pernkopf, and P. Domingos. On theoretical properties of sum-product networks. In *Proceedings of AISTATS*, pages 744–752, 2015.

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

[42] T. Rahman and V. Gogate. Merging strategies for sum-product networks: From trees to graphs. In *UAI*, 2016.

[43] A. Rashwan, P. Poupart, and C. Zhitang. Discriminative training of sum-product networks by extended Baum-Welch. In *Proceedings of PGM*, pages 356–367, 2018.

[44] A. Rashwan, H. Zhao, and P. Poupart. Online and distributed Bayesian moment matching for parameter learning in sum-product networks. In *Proceedings of AISTATS*, pages 1469–1477, 2016.

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

[46] A. Rooshenas and D. Lowd. Learning sum-product networks with direct and indirect variable interactions. *ICML – JMLR W&CP*, 32:710–718, 2014.

[47] W. Roth, R. Peharz, S. Tschiatschek, and F. Pernkopf. Hybrid generative-discriminative training of Gaussian mixture models. *Pattern Recognition Letters*, pages 131–137, 2018.

[48] O. Sharir, R. Tamari, N. Cohen, and A. Shashua. Tractable generative convolutional arithmetic circuits. *arXiv preprint arXiv:1610.04167*, 2016.

[49] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15:1929–1958, 2014.

[50] M. Trapp, R. Peharz, M. Skowron, T. Madl, F. Pernkopf, and R. Trappl. Structure inference in sum-product networks using infinite sum-product trees. In *NIPS Workshop on Practical Bayesian Nonparametrics*, 2016.

[51] A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, A. Graves, et al. Conditional image generation with pixelcnn decoders. In *Proceedings of NIPS*, pages 4790–4798, 2016.

[52] A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proceedings of ICML*, pages 1747–1756, 2016.

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

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