Conditional Sum-Product Networks: Imposing Structure on Deep Probabilistic Architectures

Xiaoting Shao* XIAOTING.SHAO@CS.TU-DARMSTADT.DE
Alejandro Molina* MOLINA@CS.TU-DARMSTADT.DE
Antonio Vergari ANTONIO.VERGARI@TUEBINGEN.MPG.DE
Karl Stelzner STELZNER@CS.TU-DARMSTADT.DE
Robert Peharz R.PEHARZ@TUE.NL
Thomas Liebig THOMAS.LIEBIG@CS.TU-DORTMUND.DE
Kristian Kersting KERSTING@CS.TU-DARMSTADT.DE

Abstract

Probabilistic graphical models are a central tool in AI, however, they are generally not as expressive as deep neural models, and inference is notoriously hard and slow. In contrast, deep probabilistic models such as sum-product networks (SPNs) capture joint distributions in a tractable fashion, but still lack the expressive power of intractable models based on deep neural networks. Therefore, we introduce conditional SPNs (CSPNs), conditional density estimators for multivariate and potentially hybrid domains that allow harnessing the expressive power of neural networks while still maintaining tractability guarantees. One way to implement CSPNs is to use an existing SPN structure and condition its parameters on the input, e.g., via a deep neural network. Our experimental evidence demonstrates that CSPNs are competitive with other probabilistic models and yield superior performance on multilabel image classification compared to mean field and mixture density networks. Furthermore, they can successfully be employed as building blocks for structured probabilistic models, such as autoregressive image models.

1. Introduction

Probabilistic models are a fundamental approach in machine learning to represent and distill meaningful representations from data with inherent structure. In practice, however, it has been challenging to come up with probabilistic models that balance three desirable goals: 1) being expressive enough to capture the complexity of real-world distributions; 2) maintain—at least on a high level—interpretable domain structure; and 3) permit a rich set of tractable inference routines.

Probabilistic graphical models (PGMs), for example, admit an interpretable structure, but are known to achieve a bad trade-off between expressivity and tractable inference (Koller and Friedman, 2009). Probabilistic models based on deep neural networks such as variational autoencoders (VAEs) (Kingma and Welling, 2014) and generative adversarial networks (GANs) (Goodfellow et al., 2014) are highly expressive, but generally lack an interpretable structure and have limited capabilities when it comes to probabilistic inference. Meanwhile, advances in deep probabilistic learning have shown that tractable models, such as arithmetic circuits (Darwiche, 2003) and sum-product networks (Poon and Domingos, 2011), can be used to capture complex distributions while maintaining a rich set of tractable inference routines.

*. Contributed Equally

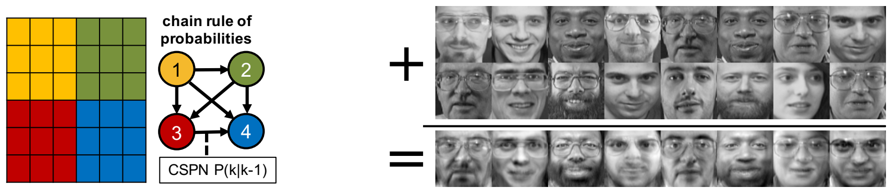

Figure 1: Imposing structure on deep probabilistic architectures. (Left) An Autoregressive Block-wise CSPN (ABCSPN) factorizes a distribution over images along image patches. (Right) Olivetti faces generated by an ABCSPN.

How can we combine these lines of research, to strive for a balance between expressivity, structure, and tractable inference? This question has received surprisingly little attention, yet. Johnson et al. (2016) combine VAEs with PGMs, and thus effectively equip expressive neural-based models with a rich structure. However, inference remains intractable in these models. Shen et al. (2019) proposed structured Bayesian networks, which impose structure among clusters of variables using probabilistic sentential decision diagrams (PSDDs) to model high-dimensional conditional distributions (Shen et al., 2018). While this yields well-structured models with a wide range of tractable inference, they are restricted to binary data and do not establish a link to deep neural networks. Moreover, PSDDs are more restricted than SPNs. Recently and independently of us, Rahman et al. (2019) proposed conditional cutset networks (CCNs), a strict sub-class of SPNs. But they do neither employ conditional independency tests for learning, nor representing mixtures of experts.

In this paper, we introduce conditional sum-product networks (CSPNs), a conditional variant of SPNs, which can harness the expressive power of universal function approximators such as neural networks, while still maintaining a wide range of inference routines. Specifically, we make the following contributions: (1) We introduce CSPNs as a deep tractable model for modelling multivariate, conditional probability distributions $P(\mathbf{Y} \mid \mathbf{X})$ over mixed variables $\mathbf{Y}$, by introducing gating nodes as mixtures with functional mixing coefficients. (2) We present a structure learning algorithm for CSPNs based on randomized conditional correlation tests (RCoT) which allows to learn structures from heterogeneous data sources. (3) We connect CSPNs with deep neural networks, and demonstrate that the improved ability of the resulting neural CSPNs to model dependencies results in increased multilabel image classification performance. (4) We illustrate how CSPNs may be used to build structured probabilistic models by introducing Autoregressive Block-wise CSPNs (ABCSPNs).

2. Conditional Probabilistic Modeling

We are interested in predicting a collection of random variables $\mathbf{Y}$ given inputs $\mathbf{X}$, i.e. we are interested in the problem of *structured output prediction* (SOP). In the probabilistic setting, SOP translates to modeling and learning a high-dimensional conditional distribution $P(\mathbf{Y} \mid \mathbf{X})$. While one could learn a univariate predictor for each variable $Y \in \mathbf{Y}$ separately, this approach assumes complete independence among $\mathbf{Y}$. This mean field assumption is often violated but still frequently used. Gaussian Processes (GPs) (Rasmussen and Williams, 2006) and Conditional Random Fields (CRFs) (Lafferty et al., 2001) are more expressive alternatives. However, they have serious shortcomings when inference has to scale to high-dimensional data or many samples. One approach to scale GPs to larger datasets is deep mixtures of GPs, introduced in (Trapp et al., 2018), which can be seen as a combination of GPs and SPNs. However, they are limited to continuous domains. In comparison, CSPNs can learn conditional distributions over heterogeneous data, i.e., where $\mathbf{Y}$ might contain discrete or continuous random variables, or even mixed data types. In a nutshell, CSPNs can tackle *unrestricted* SOP in a principled probabilistic way.

As an alternative within the family of tractable probabilistic models, logistic circuits (LCs) have been recently introduced as discriminative models (Liang and Van den Broeck, 2019), showing classification accuracy competitive to neural nets on a series of benchmarks. However, LCs and discriminative learning of SPNs (Gens and Domingos, 2012) are limited to single output prediction. Likewise, discriminative arithmetic circuits (DACs) also directly tackle modeling a conditional distribution (Rooshenas and Lowd, 2016). They are learned via compilation of CRFs, requiring sophisticated and potentially slow structure learning routines. Also related are sum-product-quotient networks (SPQNs) (Sharir and Shashua, 2017), which extend SPNs by introducing quotient nodes. This enables SPQNs to represent a conditional distribution $P(\mathbf{Y} \mid \mathbf{X})$ as the ratio $P(\mathbf{Y}, \mathbf{X})/P(\mathbf{X})$ where the two terms are modeled by two SPNs. However, CSPNs can include more complex conditional models allowing for a more compact representation.

Our neural CSPNs are close in spirit to probabilistic models based on neural networks. Generally, this line of research faces the challenge of how to parameterize distributions using the outputs of deterministic neural function approximators. Frequently, the mean field assumption is made, interpreting the output of the network as the parameters of primitive univariate distributions, assuming independence among random variables. Modelling complex distributions must then involve sampling as in VAEs (Kingma and Welling, 2014) or hierarchical variational models (Ranganath et al., 2016), causing significant computational overhead and yielding highly intractable models. Other approaches for conditional density estimation based on neural networks include (conditional) normalizing flows, which yield tractable likelihoods, but are limited to continuous distributions and come with significant computational costs for computing the determinant of the Jacobian (Rezende and Mohamed, 2015). Generally, CSPNs are most closely related to two classic approaches, namely mixture density networks (MDNs) (Bishop, 1994) and (hierarchical) mixtures of experts (Jordan and Jacobs, 1994). These models use the output of neural networks to parameterize a (typically Gaussian) mixture model. Shallow mixture models, however, are often inadequate in high dimensions, as the number of components required to accurately model the data may grow exponentially with the number of dimensions. SPNs address this by encoding a hierarchy of mixtures. Neural CSPNs may consequently be seen as a deep, hierarchical version of MDNs and MoEs.

3. (Unconditional) Sum-Product Networks

Here, we briefly review classical (unconditional) SPNs. For details, see (Darwiche, 2003; Poon and Domingos, 2011; Peharz et al., 2017). We denote random variables (RVs) as upper-case letters, e.g., V, their values as lower-case letters, e.g., $v \sim V$; and sets of RVs in bold, e.g., $\mathbf{v} \sim \mathbf{V}$.

A sum-product network (SPN) over a set of random variables $\mathbf{V}$ is defined via an acyclic directed graph (DAG) containing three types of nodes: distribution nodes, sum nodes and product nodes. All leaves of the DAG are distribution nodes, and all internal nodes are either sums or products. An SPN leaf represents a univariate distribution $P(Y)$ for some RV $Y \in \mathbf{V}$. A sum node represents a *mixture* $\sum_k w_k P_k(\mathbf{Y})$, where P_k are the children of the sum node according to the DAG, and w_k satisfy $w_k \geq 0$ and $\sum_k w_k = 1$. In comparison to classical probabilistic graphical

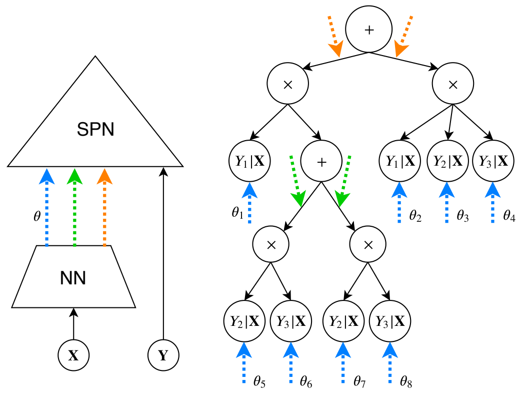

Figure 2: Overview of the architecture (left) and a concrete CSPN example encoding $P(\mathbf{Y} \mid \mathbf{X})$ (right). $\mathbf{X}$ is the set of conditional variables and $\mathbf{Y}$ consists of Y_1, Y_2 and Y_3. Each color of the arrow represents one data flow. The gating weights, possibly also the leaves, are parameterized by the output of a neural network given $\mathbf{X}$.

models, SPNs have the advantage that they can represent certain distributions much more succinct, even if the corresponding graphical model would have very high tree-width (Koller and Friedman, 2009). Furthermore, *inference routines* such as *marginalization* and *conditioning* can be tackled in linear time w.r.t. the network size (Darwiche, 2003; Poon and Domingos, 2011). These tasks can also be *compiled*, i.e., starting from an SPN representing $P(\mathbf{V})$, it is possible to generate a new SPN representing the marginal distribution for an arbitrary $\mathbf{X} \subset \mathbf{V}$, potentially conditioned on event $\mathbf{Y} = \mathbf{y}$, for any $\mathbf{Y} \subset \mathbf{V}$, $\mathbf{Y} \cap \mathbf{X} = \emptyset$.

Indeed, one way to represent conditional distribution $P(\mathbf{Y} \mid \mathbf{X})$ is to train a regular SPN on the joint $P(\mathbf{V})$, and to then compile it into a conditional distribution for each input $\mathbf{x}$ of interest. It is clear, however, that this will generally deliver sub-optimal results, since the network will not be specialized to the specific input-output pair of $\mathbf{X}$ and $\mathbf{Y}$. Intuitively, training the full joint $P(\mathbf{V})$ optimizes *all* possible conditional distributions one can derive from the joint. Thus, when we are interested in learning a conditional distribution for a set of input and output variables $\mathbf{X}$ and $\mathbf{Y}$ known a-priori, directly learning a conditional distribution can be expected to deliver better results.

4. Conditional Sum-Product Networks

We now introduce a notion of conditional SPNs (CSPNs), which structurally reflect conditional independencies, and propose a learning framework to induce CSPNs from data. To this end, we employ $\mathbf{Y} \subset \mathbf{V}$ to denote the target RVs, also called labels, while we denote the disjoint set of observed RVs, also called features, as $\mathbf{X} := \mathbf{V} \setminus \mathbf{Y}$.

Definition of Conditional SPNs (CSPNs). We define a CSPN as a rooted DAG containing three types of nodes, namely *leaf*, *gating*, and *product* nodes, encoding a conditional probability distribution $P(\mathbf{Y} \mid \mathbf{X})$. See Fig. 2 for an illustrative example of a CSPN. Each leaf encodes a normalized univariate conditional distribution $P(Y \mid \mathbf{X})$ over a target RV $Y \in \mathbf{Y}$, where Y is denoted as the leaf's *conditional scope*. A product node factorizes a conditional probability distribution over its children, i.e., $\prod_k P_k(\mathbf{Y}_k \mid \mathbf{X})$ where $\mathbf{Y}_k \subset \mathbf{Y}$. To encode functional dependencies on the input features, a *gating* node computes $\sum_k g_k(\mathbf{X}) P_k(\mathbf{Y} \mid \mathbf{X})$ where g_k is the output of a non-negative function g w.r.t. the k-th child node, such that $\sum_k g_k(\mathbf{X}) = 1$. One may also choose a constant gating function and the gating node hence becomes a standard sum node. The conditional scope of a non-leaf node is the union of the conditional scopes of its children.

Gating nodes are akin to gates in *mixtures of experts* (Bishop et al., 1995; Shazeer et al., 2017), which motivates the name. The notions of completeness and decomposability of SPNs (Poon and

Domingos, 2011) naturally carry over to CSPNs, and we can reuse the efficient SPN inference routines, guaranteeing that any conditional marginal probability may be computed exactly in linear time (Rooshenas and Lowd, 2016). This can easily be verified by considering that for a fixed input $\mathbf{x}$, a CSPN reduces to a standard SPN over the labels $\mathbf{Y}$, allowing any inference routine for standard SPNs (Poon and Domingos, 2011; Peharz et al., 2015; Vergari et al., 2015) to be employed. However, CSPNs are more powerful than SPNs.

CSPNs are More Powerful than SPNs. Neural networks are known to be universal approximators. As Choi and Darwiche (2018) argue, a probabilistic model[1] over finite-discrete random variables represents not only one function, but a function for each possible probabilistic query — the (conditional) marginals. Unfortunately, the functions corresponding to queries, unlike neural networks, are generally not universal approximators. They are restricted to multi-linear functions of primitive distributions or quotients thereof. Consequently, some functions may only be (approximately) fitted using a large number of e.g. Gaussian leaf distributions. By adding gating nodes, however, CSPNs extend SPNs in a way that allows them to induce universal approximators. This can be seen as follows. Using threshold functions $x_i \leq c$ ($c \in \mathbb{R}$) as gates, one can encode *testing arithmetic circuits* (Choi and Darwiche, 2018) as CSPNs. Testing arithmetic circuits, however, have been proven to encode piece-wise multi-linear functions and in turn ResNets, which are universal approximators. Moreover, the class of Gaussian CSPNs mean functions is dense in the class of all continuous functions over arbitrary compact domains. To see this, we note that one can use single-output softmax gates in order to facilitate modelling mixtures of experts models $\sum_k g_k(\mathbf{X}) P_k(\mathbf{Y} \mid \mathbf{X})$ in the form of Nguyen *et al.* (2016). Consequently, their result carries over.

5. Learning Conditional Sum-Product Networks

A simple way to construct CSPNs representing $P(\mathbf{Y} \mid \mathbf{X})$ is to start from a standard or even random SPN over $\mathbf{Y}$, and to make its parameters (i.e., the sum weights and parameters of leaf distributions) functional of input $\mathbf{x}$, defining $P(\mathbf{Y} \mid \mathbf{X} = \mathbf{x}) := P(\mathbf{Y}; \theta)$, where parameters $\theta := g(\mathbf{x})$ are a function of the input $\mathbf{x}$. For g we might use arbitrary function representations, such as deep neural networks. This architecture, which we call *neural* CSPN, works very well, as we demonstrate in the experimental section. However, we may fail to exploit some conditional independencies among the random variables. Consequently, we now introduce a structure learning strategy LearnCSPN extending the established LearnSPN algorithm (Gens and Domingos, 2013) which has been instantiated several times for learning (unconditional) SPNs under different distributional assumptions (Vergari et al., 2015; Molina et al., 2018).

Our LearnCSPN routine builds a CSPN in a top-down fashion by introducing nodes while partitioning a data matrix in a recursive and greedy manner. It creates one of the three node types at each step — (1) a leaf, (2) a product, or (3) a gating node. If a single target RV Y is present, one conditional probability distribution can be fit as a leaf. To generate product nodes, conditional independencies are found by means of a statistical test to partition the set of target RVs $\mathbf{Y}$. If no such partitioning is found, then training samples are partitioned into clusters (conditioning) to induce a gating node. Finally, to calibrate all parameters, capturing cross-covariances between $\mathbf{Y}$ and $\mathbf{X}$, we run an end-to-end parameter estimation. Let us now review the three steps of LearnCSPN more in detail.

1. In their exposition, they focus on Bayesian networks and arithmetic circuits, but their arguments carry over to SPNs.

(1) Inducing Leaves. In order to allow for tractable inference, we require conditional models at the leaves to be normalized. Apart from this requirement, *any* such univariate tractable conditional model may be plugged in a CSPN effortlessly to model $P(Y \mid \mathbf{X})$. This can be simple univariate models or the joint output of a deep model. For the sake of simplicity, we here use Generalized Linear Models (GLMs) (McCullagh, 1984) as leaves, due to their simple and flexible nature. We compute $P(Y \mid \mu)$ with $\mu := \mathrm{glm}(\mathbf{X})$ by regressing univariate parameters μ from features $\mathbf{X}$, for a given set of distributions in the exponential family.

(2) Inducing Product Nodes. For product nodes, we are interested in decomposing the labels $\mathbf{Y}$ into conditionally-independent subsets via conditional independence (CI) tests. In terms of density functions, testing that Y_i is independent of Y_j given $\mathbf{X} = \mathbf{x}$, for any value of $\mathbf{x}$, i.e., $Y_i \perp\!\!\!\perp Y_j | \mathbf{X}$, can equivalently be characterized as $P(Y_i, Y_j \mid \mathbf{X}) = P(Y_i \mid \mathbf{X}) P(Y_j \mid \mathbf{X})$. To accommodate arbitrary conditional distributions at the leaves, regardless of their parametric likelihood models, we adopt a non-parametric pairwise CI test procedure to decompose labels $\mathbf{Y}$. Specifically, the randomized conditional correlation test (RCoT) (Strobl et al., 2019) is used. It computes the squared Hilbert-Schmidt norm of the partial cross-covariance operator and uses the Lindsay-Pilla-Basak method to approximate the asymptotic distribution. We then create a graph where the nodes are RVs in $\mathbf{Y}$ and put an edge between two nodes Y_i, Y_j if we cannot reject the null hypothesis that $Y_i \perp\!\!\!\perp Y_j | \mathbf{X}$ for a given threshold α. The conditional scopes of product children are then given by the connected components of this graph, akin to (Gens and Domingos, 2013).

(3) Inducing Gating Nodes. A probabilistic interpretation of a gating node can be given in the context of mixture models for conditional probability distributions $\sum_k g_k(\mathbf{X}) P_k(\mathbf{Y} \mid \mathbf{X})$ (Bishop et al., 1995). Estimating them is a form of *conditional clustering* (He et al., 2017) to accommodating for known cross-covariates between $\mathbf{Y}$ and $\mathbf{X}$. Here, we approach this by inducing the gating node structure using clustering of the features $\mathbf{X}$ only and calibrating the cross-covariates afterwards by end-to-end parameters estimation. The clustering scheme can be instantiated based on the available knowledge of the data distribution (e.g., k-Means for Gaussians); one can also leverage random splits, as in random projection trees (Dasgupta and Freund, 2008). Then, we select a functional form for the gating function $g_k(\mathbf{X})$, ideally, a differentiable parametric one such as logistic regression or a (deep) neural network with a softmax layer such that constraints of a proper mixture of distributions, i.e., $\sum_k g_k(\mathbf{X}) = 1$ and $\forall_{\mathbf{X}} g_k(\mathbf{X}) >= 0$ are fulfilled. We denote the corresponding cluster assignment as a one-hot coded vector $\mathbf{Z}$ and fit the gating function to predict $\mathbf{Z}_k = g_k(X)$.

(4) Calibrating Cross-Covariances. Given the induced structure of the complete CSPN, we calibrate the cross-covariances between $\mathbf{X}$ and $\mathbf{Y}$ by final joint parameter estimation in an end-to-end fashion. That is, we fit each gating function to predict $\mathbf{Z}_k := \arg\max g_k(\mathbf{X})$ (using e.g. softmax) and the parameters of the leaf distributions jointly. This is a proper generalization of traditional sum nodes in SPNs: sum node is a special case of a gating node where the mixtures are constants independent of $\mathbf{X}$, i.e. $g_k(\mathbf{X}) = c$. Moreover, this functional mixing does not break the tractability guarantees over $\mathbf{Y}$ as $\mathbf{X}$ is always assumed to be evidence in the conditional case.

To summarize, LearnCSPN automatically sets the parameters of the gating function as described above. The parameters of the GLMs are obtained by an Iteratively Reweighted Least Squares (IRWLS) algorithm as described in (Green, 1984), on the instances available at the *leaf* node. However, those parameters are locally optimized and usually not optimal for the global distribution. Fortunately, CSPNs are differentiable as long as the leaf models and gating functions are differentiable. Hence, one can optimize the conditional likelihood in an end-to-end fashion using gradient-based optimization techniques.

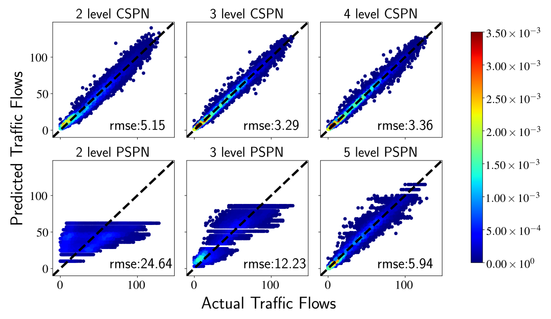

Figure 3: CSPNs can encode nontrivial conditional distributions: Traffic flow predictions of Poisson CSPNs (top) versus SPNs (botton) for shallow (left) or deep models (center and right). CSPNs are consistently more accurate than corresponding SPNs and, as expected, deeper CSPNs outperform shallow ones (center and right).

6. Imposing Structure on Deep Generative Models: Autoregressive Block-wise CSPN

To illustrate how to impose structure on generative models by employing CSPNs as building blocks, we construct an autoregressive model composed of CSPNs. This is basically done by representing a joint distribution as a factorization of conditional models, as in Bayesian networks. Specifically, by applying the chain rule of probabilities, we can decompose a joint distribution as the product $P(\mathbf{Y}, \mathbf{X}) = P(\mathbf{Y} \mid \mathbf{X})P(\mathbf{X})$. Then, one could learn an SPN to model $P(\mathbf{X})$ and a CSPN for $P(\mathbf{Y} \mid \mathbf{X})$. By combining both models using a single product node, we represent the whole joint as a computational graph. Now, if one applies the same operation several times by repeatedly partitioning $\mathbf{Y}$ in a series of disjoint sets $\mathbf{Y}_1, \mathbf{Y}_2, \ldots$ we can obtain an *autoregressive* model representation, which we call an *Autoregressive Block-wise CSPN* (ABCSPN). Inspired by image autoregressive models like PixelCNN (van den Oord et al., 2016a) and PixelRNN (van den Oord et al., 2016b), we propose the ABCSPN for conditional image generation to illustrate the benefits of imposing structure. For one ABCSPN, we divide images into pixel blocks, hence factorizing the joint distribution block-wise instead of pixel-wise as in PixelC/RNN. Each factor accounting for a block of pixels is then a CSPN representing the distribution of those pixels as conditioned on all previous blocks and on the class labels, cf. Fig. 1 (left).[2] We factorize blocks in raster scan order, row by row and left to right, but any other ordering is also possible. The complete generative model over image $\mathbf{I}$ encodes: $p(\mathbf{I}) = \prod_{i=1}^{n} p(\mathbf{B}_i \mid \mathbf{B}_1, \ldots, \mathbf{B}_{i-1}, \mathbf{C}) \cdot p(\mathbf{C})$ where $\mathbf{B}_i$ denotes the pixel RVs of the i-th block and $\mathbf{C}$ the one-hot coded image class. Learning each conditional block as a CSPN can be done by the structure learning routine introduced above.

7. Empirical Evidence

Here we investigate CSPNs in experiments on real-world data. Specifically, we aim to answer the following questions: **(Q1)** Can CSPNs perform better than regular SPNs? **(Q2)** How accurate are CSPNs for SOP? **(Q3)** How do ABCSPNs perform on real data? **(Q4)** Do neural CSPNs outperform baseline neural conditional models such as MDNs on image SOP tasks? To this end, we implemented CSPNs in Python calling TensorFlow and R.

(Q1, Q2) Multivariate Traffic Data Prediction. We employ CSPNs for multivariate traffic data prediction, comparing them against SPNs with Poisson leaf distributions (Molina et al., 2017). This is an appropriate model as the traffic data represents counts of vehicles. We considered temporal vehicular traffic flows in the German city of Cologne (Ide et al., 2015). The data comprises 39 RVs

2. Here, image labels play the role of observed RVs in $\mathbf{X}$.

Table 1: Average test conditional log-likelihood (CLL) of DACL, CCNs and CSPNs on 20 standard density estimation benchmarks. Best results are bold. As the numbers of wins for DACL, CCNs and CSPNs show, CSPNs are competitive to state-of-the-art.

		Nltcs	Msnbc	KDD	Plants	Audio	Jester	Netflix	Accidents	Retail	Pumsb.
50% evidence	DACL	-2.770	-2.918	**-0.998**	-4.655	-18.958	-24.830	-26.245	**-9.718**	**-4.825**	**-6.363**
	CCN	**-2.58**	**-2.18**	-1.19	**-4.53**	-18.67	-24.96	-26.03	-10.24	-4.88	-6.98
	CSPN	-2.795	-3.165	-1.023	-4.720	**-18.543**	**-24.543**	**-25.914**	-11.587	-5.600	-7.383
80% evidence	DACL	-1.255	-1.557	-0.386	-1.812	-7.337	-9.998	-10.482	**-3.493**	-1.687	-2.594
	CCN	**-0.99**	**-0.87**	**-0.36**	**-1.54**	-7.58	**-9.75**	**-10.22**	-3.61	-1.66	**-2.02**
	CSPN	-1.256	-1.684	-0.397	-1.683	**-7.110**	-9.830	-10.351	-4.045	**-1.654**	-2.618
		Dna	Kosarek	MSWeb	Book	EachMovie	WebKB	Reuters-52	20News	Bbc	Ad
50% evidence	DACL	-34.737	-5.053	-5.653	-16.801	-23.325	-72.072	-41.544	-76.063	-118.684	-4.893
	CCN	**-32.98**	**-4.76**	**-4.25**	-15.90	-24.85	-69.34	-36.56	-71.69	-114.52	**-3.41**
	CSPN	-38.243	-5.527	-6.686	**-10.653**	**-18.130**	**-18.542**	**-15.736**	**-35.900**	**-47.138**	-6.290
80% evidence	DACL	-12.116	-2.549	**-1.333**	-6.817	-9.403	-28.087	-17.143	-27.918	-44.811	-1.370
	CCN	-12.29	**-1.14**	-1.69	-6.48	-8.40	-25.76	-15.67	-27.72	-43.27	-1.18
	CSPN	**-11.895**	-2.397	-1.335	**-3.191**	**-4.579**	**-2.623**	**-3.878**	**-4.984**	**-2.996**	**-1.030**
50% evidence	**Wins**	DACL: 4	CCN: 7	CSPN: **9**							
80% evidence	**Wins**	DACL: 2	CCN: 8	CSPN: **10**							

whose values are from stationary detectors located at the 50km long Cologne orbital freeway, each one counting the number of vehicles within a fixed time interval. It contains 1440 samples, each of which is a snapshot of the traffic flow. The task of the experiments is to predict the next snapshot ($|\mathbf{Y}| = 39$) given a historical one ($|\mathbf{X}| = 39$).

We trained both CSPNs and SPNs controlling the depth of the models. The CSPNs use GLMs with exponential link function as Poisson univariate conditional leaves. Results are summarized in Fig. 3. We can see that CSPNs are always the most accurate model as their root mean squared error (RMSE) is always the lowest. As expected, deeper CSPNs have lower predictive error compared to shallow CSPNs. Moreover smaller CSPNs perform equally well or even better than SPNs, empirically confirming that CSPN are more expressive than SPNs. This answers **(Q1, Q2)** affirmatively and also provides evidence for the convenience of directly modeling a conditional distribution.

(Q2) Conditional Density Estimation. We now focus on conditional density estimation on 20 standard binary benchmark datasets. The number of variables of these datasets range from 16 to 1556. In order to estimate conditional density, features are splitted into evidences ($|\mathbf{X}|$) and targets ($|\mathbf{Y}|$) with different proportions[3]. We compare to DACL (Rooshenas and Lowd, 2016) and CCNs (Rahman et al., 2019) as they currently provide state-of-the-art conditional log-likelihoods (CLLs) on such data. To this end, we first perform structure learning on the train data split (stopping learning when no more than 10% of samples are available), followed by end-to-end learning on the train and validation data. Note that the sophisticated structure learning in DACL directly optimizes for the CLL at each iteration.

Results are reported in Tab. 1 (best in bold). Since we do not have access to per-instance CLLs on CCNs, t-test is not possible, but the results still provide a tendency for comparison. We can see that for both the 80%-evidence scenario and the 50%-evidence scenario, CSPNs win the most. In total, CSPNs perform on par with CCNs (wins are quite balanced) and outperform DACL. Besides,

3. We adopted the data splits of Rooshenas and Lowd (2016).

we note that CSPNs are faster to learn than DACL and that, in practice, no real hyperparameter tuning was necessary to achieve these scores, while DACL ones are the result of a fine grained grid search (see (Rooshenas and Lowd, 2016)). This answers **(Q2)** affirmatively and shows that CSPNs are comparable to state-of-the-art.

(Q3) Auto-Regressive Image Generation. We investigate ABCSPNs on Olivetti faces by splitting each image into 16 resp. 64 blocks of equal size. Then we trained a CSPN on Gaussian domain for each block conditioned on all the blocks above and to the left of it and on the image class and formulate the distribution of the images as the product of all the CSPNs. In Fig. 1 (right), new faces are sampled from an ABCSPN after conditioning on a set of class images that is the mixing of two original classes in the Olivetti dataset. That is, by conditioning on multiple classes it generates samples that resemble both individuals belonging to those classes, even though the ABCSPN never saw that class combination before during training. As one can see, these samples from ABCSPNs look very plausible. ABCSPNs achieve this while reducing the number of independency tests among pixels required by CSPNs: from quadratic over all pixels in an image down to quadratic in the block size. This demonstrates how ABCSPNs are able to learn meaningful and accurate models over the image manifold, providing an affirmative answer to **(Q3)**.

(Q4) Neural CSPNs with Random Structures. In high-dimensional domains, such as images, the structure learning procedure introduced above may be intractable. In this case, CSPNs may still be applied by starting from a random SPN structure as Peharz et al. (2019) proposed, resulting in a flexible distribution $P(\mathbf{Y}; \theta := g(\mathbf{X}))$. When g is represented by a deep neural network, we obtain a highly expressive conditional density estimator which can be trained end-to-end. To demonstrate the efficacy of this approach, we evaluate it on several multilabel image classification tasks. The goal of each task is to predict the joint conditional distribution of binary labels $\mathbf{Y}$ given an image $\mathbf{X}$. We experiment on the CelebA dataset, which features images of faces annotated with 40 binary attributes. In addition, we constructed multilabel versions of the MNIST and Fashion-MNIST datasets, by adding additional labels indicating symmetry, size, etc. to the existing class labels, yielding 16 binary labels total.

We compare our model to two different common ways of parameterizing conditional distributions using neural networks. The first is the mean field approximation, whereby the output of a neural network is interpreted as logits of independent univariate Bernoulli distributions, assuming that the labels $\mathbf{Y}$ are conditionally independent given $\mathbf{X}$. Second, we compare to mixture density networks with 10 mixture components, each itself a mean field distribution. For each of these models, including the CSPN, we use the same standard convolutional neural network architecture up to the last two layers. Those final two layers are customized to the different desired output formats: For the mean field and MDN models, all parameters are predicted using two dense layers. For the CSPN, we use a dense layer followed by a 1d-convolution, in order to obtain the increased number of SPN parameters without using drastically more neural network weights. The resulting conditional log-likelihoods as well as accuracies are given in Table 2. To compute accuracies, we obtained MPE estimates from the models using the standard max-product approximation. On the MNIST and Fashion dataset, estimates were counted as accurate only if all 16 labels were correct, on the CelebA dataset, we report the average accuracy across all 40 labels. The results indicate that the mean field approximation is inappropriate on the considered datasets, as allowing the inclusion of conditional dependencies resulted in a pronounced increase in both likelihood and accuracy. The improved model capacity of the CSPN compared to the MDN yielded a further performance

Table 2: Comparison to mean field models and mixture density networks: Average test conditional log-likelihood (CLL) and test accuracy of the mean field (MF) model, mixture density network (MDN), and neural conditional SPN (CSPN) on multilabel image classification tasks. The best results are marked in bold. As one can see, the additional representational power of CSPNs yields notable improvements.

	CLL			ACCURACY		
	MF	MDN	CSPN	MF	MDN	CSPN
MNIST	-0.70	-0.61	**-0.54**	74.1%	76.4%	**78.4%**
FASHION	-0.95	-0.73	**-0.70**	73.4%	73.7%	**75.5%**
CELEBA	-12.1	-11.6	**-10.8**	86.6%	85.3%	**87.8%**

increase. On CelebA, CSPN outperforms a number of sophisticated neural network architectures, despite being based on a standard convnet with only about 400k parameters (Ehrlich et al., 2016).

8. Conclusions

We extended the concept of sum-product networks (SPNs) towards conditional distributions by introducing conditional SPNs (CSPNs). Conceptually, they combine simpler models in a hierarchical fashion in order to create a deep representation that can model multivariate and mixed conditional distributions while maintaining tractability. They can be used to impose structure on deep probabilistic models and, in turn, significantly boost their power as demonstrated by our experimental results. Much remains to be explored, including other learning methods for CSPNs, design principles for CSPN+SPN architectures, combining the (C)SPN stack with the deep neural learning stack, more work on extensions to sequential and autoregressive domains, and further applications.

Acknowledgements. The authors acknowledge the support of the German Science Foundation (DFG) project ”Argumentative Machine Learning” (CAML, KE 1686/3-1) of the SPP 1999 “Robust Argumentation Machines” (RATIO). KK also acknowledges the support of the Rhine-Main Universities Network for “Deep Continuous-Discrete Machine Learning” (DeCoDeML), and TL the support of the DFG under project B4 ‘Analysis and Communication for Dynamic Traffic Prognosis’ of the Collaborative Research Centre SFB 876. This project has also received funding from the European Union’s Horizon 2020 research and innovation programme under the Marie Skłodowska-Curie Grant Agreement No. 797223.

References

C. M. Bishop. Mixture density networks. Technical Report NCRG/4288, Aston University, Birmingham, UK, 1994.

C. M. Bishop et al. *Neural networks for pattern recognition.* Oxford university press, 1995.

A. Choi and A. Darwiche. On the relative expressiveness of bayesian and neural networks. In *Proc. of PGM*, volume 72, pages 157–168. PMLR, 2018.

A. Darwiche. A differential approach to inference in Bayesian networks. *Journal of ACM*, 2003.

S. Dasgupta and Y. Freund. Random projection trees and low dimensional manifolds. In *Proc. of Symp. Theory of computing*, 2008.

M. Ehrlich, T. J. Shields, T. Almaev, and M. R. Amer. Facial attributes classification using multi-task representation learning. In *Proc. of the CVPR Workshops*, pages 47–55, 2016.

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

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

I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proc. of NIPS*, 2014.

P. J. Green. Iteratively reweighted least squares for maximum likelihood estimation, and some robust and resistant alternatives. *JSTOR*, 1984.

X. He, T. Gumbsch, D. Roqueiro, and K. Borgwardt. Kernel conditional clustering. In *2017 IEEE International Conference on Data Mining (ICDM)*, pages 157–166. IEEE, 2017.

C. Ide, F. Hadiji, L. Habel, A. Molina, T. Zaksek, M. Schreckenberg, K. Kersting, and C. Wietfeld. Lte connectivity and vehicular traffic prediction based on machine learning approaches. In *VTC*. IEEE, 2015.

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

M. I. Jordan and R. A. Jacobs. Hierarchical mixtures of experts and the em algorithm. *Neural computation*, 6(2):181–214, 1994.

D. P. Kingma and M. Welling. Auto-encoding variational Bayes. In *Proc. of ICLR*, 2014.

D. Koller and N. Friedman. *Probabilistic Graphical Models: Principles and Techniques*. MIT Press, 2009. ISBN 0-262-01319-3.

J. D. Lafferty, A. McCallum, and F. C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proc. of ICML*, pages 282–289, 2001.

Y. Liang and G. Van den Broeck. Learning logistic circuits. In *Proc. of AAAI*, 2019.

P. McCullagh. Generalized linear models. *EJOR*, 16(3), 1984.

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

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 *Proc. of AAAI*, 2018.

H. D. Nguyen, L. R. Lloyd-Jones, and G. J. McLachlan. A universal approximation theorem for mixture-of-experts models. *Neural Comput.*, 28(12):2585–2593, 2016. ISSN 0899-7667.

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

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

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

H. Poon and P. Domingos. Sum-Product Networks: a New Deep Architecture. *Proc. of UAI*, 2011.

T. Rahman, S. Jin, and V. Gogate. Cutset bayesian networks: A new representation for learning rao-blackwellised graphical models. In *Proc. of IJCAI*, 2019.

R. Ranganath, D. Tran, and D. Blei. Hierarchical variational models. In *Proc. of ICML*, 2016.

C. E. Rasmussen and C. K. I. Williams. *Gaussian processes for machine learning*. Adaptive computation and machine learning. MIT Press, 2006.

D. J. Rezende and S. Mohamed. Variational inference with normalizing flows. In *Proc. of ICML*, pages 1530–1538, 2015.

A. Rooshenas and D. Lowd. Discriminative structure learning of arithmetic circuits. In *Proc. of AISTATS*, 2016.

O. Sharir and A. Shashua. Sum-product-quotient networks. *Proc. of AISTATS*, 2017.

N. Shazeer, A. Mirhoseini, K. Maziarz, A. Davis, Q. Le, G. Hinton, and J. Dean. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *Proc. of ICLR*, 2017.

Y. Shen, A. Choi, and D. A. Conditional PSDDs: Modeling and learning with modular knowledge. In *Proc. of AAAI*, 2018.

Y. Shen, A. Goyanka, A. Darwiche, and A. Choi. Structured bayesian networks: From inference to learning with routes. *Proc. of AAAI*, 33(01):7957–7965, 2019.

E. V. Strobl, K. Zhang, and S. Visweswaran. Approximate kernel-based conditional independence tests for fast non-parametric causal discovery. *Journal of Causal Inference*, 7(1), 2019.

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

A. van den Oord, N. Kalchbrenner, L. Espeholt, O. Vinyals, et al. Conditional image generation with pixelcnn decoders. In *Proc. of NIPS*, 2016a.

A. van den Oord, N. Kalchbrenner, and K. Kavukcuoglu. Pixel recurrent neural networks. In *Proc. of ICML*, 2016b.

A. Vergari, N. Di Mauro, and F. Esposito. Simplifying, regularizing and strengthening spn structure learning. In *Proc. of ECML/PKDD*, 2015.