Conditionally Tractable Density Estimation using Neural Networks

Hailiang Dong, Chiradeep Roy, Tahrima Rahman, Vibhav Gogate, Nicholas Ruozzi
The University of Texas at Dallas, Richardson, TX, 75080, USA
{ *hailiang.dong, chiradeep.roy, tahrima.rahman, vibhav.gogate, nicholas.ruozzi*}*@utdallas.edu*

Abstract

Tractable models such as cutset networks and sum-product networks (SPNs) have become increasingly popular because they have superior predictive performance. Among them, cutset networks, which model the mechanics of Pearl's cutset conditioning algorithm, demonstrate great scalability and prediction accuracy. Existing research on cutset networks has mainly focused on discrete domains, and the best mechanism to extend cutset networks to continuous domains is unclear. We propose one possible alternative to cutset networks that models the full joint distribution as the product of a local, complex distribution over a small subset of variables and a fully tractable conditional distribution whose parameters are controlled using a neural network. This model admits exact inference when all variables in the local distribution are observed, and although the model is not fully tractable in general, we show that "cutset" sampling can be employed to efficiently generate accurate predictions in practice. We show that our model performs comparably or better than existing competitors through a variety of prediction tasks on real datasets.

1 INTRODUCTION

Tractable probabilistic models such as cutset networks (Rahman et al., 2014), sum-product networks (Poon and Domingos, 2011), arithmetic circuits (Darwiche, 2003), and probabilistic sentential decision diagrams (Kisa et al., 2014) have gained popularity in recent years, primarily because of their ability to accurately answer various reasoning queries in linear time in the size of the model while maintaining other desirable properties such as high expressivity and flexibility. However, to date, with a few exceptions (Poon and Domingos, 2011; Molina et al., 2018; Uria et al., 2013), a majority of work on learning tractable models has focused on discrete random variables. This is primarily due to the fact that, unlike discrete domains, guaranteeing tractability of models in continuous domains while also maintaining their high expressivity and flexibility is very challenging.

Existing tractable models for continuous variables operate under the assumption that the underlying data can be modeled using a mixture of parametric (e.g., Gaussian) or non-parametric probability distributions. For instance, sum-product networks (SPNs) (Poon and Domingos, 2011) and mixed-SPNs (Molina et al., 2018) use latent discrete (variable) architectures distributed over sum and product nodes via the *distributive law* to efficiently represent a Gaussian and non-parametric mixture respectively. Our goal in this paper is to move beyond sum-product mixtures and develop a modeling framework for continuous domains that can effectively handle non-linear dependencies and still admit efficient inference schemes.

To this end, we propose a novel tractable model for continuous domains motivated by cutset networks (CNs) (Rahman et al., 2014). CNs and their subsequent extensions (Rahman and Gogate, 2016; Roy et al., 2021) make use of cutset conditioning (Pearl, 1988): select a set of variables to condition on (namely cutset variables), such that the distribution over the remaining variables (leaf variables) can be well-represented by tractable, tree-structured Bayesian networks (BNs) (Chow and Liu, 1968). Cutset networks are expressive and have shown great scalability and predictive accuracy in high-dimensional discrete domains. A natural extension of cutset networks to continuous domains is to condition each cutset variable over a finite number of ranges, similar to decision tree regressors, and then model the leaf distributions using conditional linear Gaussian (CLGs) Bayesian networks (Grzegorczyk, 2010). Although simple and appealing, the

Proceedings of the 25$^{\text{th}}$ International Conference on Artificial Intelligence and Statistics (AISTATS) 2022, Valencia, Spain. PMLR: Volume 151. Copyright 2022 by the author(s).

number of cutset variables should be small enough to guarantee tractability but large enough not to incur a significant loss in expressive power. Further, while cutset networks condition on all possible values of a discrete variable, in the continuous case, conditioning will correspond to partitioning the space into rectangular blocks (hyper cubes) – the leaves are then distributions over rectangular regions, which is not natural in many practical applications.

To overcome these limitations of cutset networks in continuous domains, and motivated by the work of Rahman et al. (2019) on cutset Bayesian networks, we propose to model the joint distribution as a product of two distributions: (1) a local, complex unconditional distribution defined over a small subset of variables $\mathbf{X}$ and (2) a fully tractable conditional distribution defined over the remaining variables $\mathbf{Y}$ whose parameters are controlled by a neural network. Like cutset Bayesian networks, this model admits exact inference when all variables in the set $\mathbf{X}$ are observed, and although the model is not fully tractable, we show that "cutset" sampling can be employed to efficiently generate accurate predictions in practice. We demonstrate the competitive performance of our approach against two well-known models in the literature: SPNs (Poon and Domingos, 2011; Molina et al., 2018), which use latent discrete sum-product architectures and RNADE (Uria et al., 2016), which uses neural density estimators, using two sets of prediction tasks on real datasets.

To summarize our contributions, we present (1) a novel continuous tractable architecture, inspired by cutset networks, (2) a novel neural network architecture for choosing the parameters of Gaussian Bayesian Network, (3) and an extensive study to evaluate the performance of our model, SPNs, and RNADE through various prediction tasks on real datasets.

2 RELATED WORK

There is a large body of work on modeling large, continuous multi-variate distributions using parametric, semi-parametric, and non-parametric representations; see e.g., (Lauritzen and Wermuth, 1989; Hofmann and Tresp, 1996; Monti and Cooper, 1998; Lauritzen, 1996). More recently, there has been growing interest in learning tractable continuous models using the sum-product network framework (Poon and Domingos, 2011; Molina et al., 2018; Darwiche, 2003). These models can compactly represent sophisticated mixtures of parametric or non-parametric distributions using a DAG. Moreover, one can solve many standard inference tasks such as posterior marginal inference in time that scales linearly with the size of the DAG.

Another popular approach is to employ neural models to build density estimators (Monti and Cooper, 1998; Uria et al., 2013, 2016; Papamakarios et al., 2017; Germain et al., 2015). Although, these models yield good test set loglikelihood scores, they typically have poor prediction accuracy because probabilistic inference over them is intractable.

In this work, we aim to develop a novel modeling framework for continuous domains which enables the user to learn models that are intractable in general but admit efficient cutset-based inference algorithms (Bidyuk and Dechter, 2007). We employ neural networks for computing the parameters of our proposed model similar to Jordan and Jacobs (1994); Shao et al. (2020) and Thoma et al. (2021). However, unlike these previous works, which learn discriminative models, we learn generative models. Moreover, we use a different template distribution for the underlying model and develop a novel neural network architecture for predicting the parameters of the model.

3 CONDITIONALLY TRACTABLE DENSITY ESTIMATION

In this section, we describe our approach for building tractable models in continuous domains. Note that we use bold uppercase letters for a set of random variables, e.g., $\boldsymbol{X}$, while a single random variable is denoted using uppercase letters, e.g., A. The instantiations (configurations) of random variables are denoted as lowercase letters. For example, $\boldsymbol{x}$ is one possible configuration for all variables in $\boldsymbol{X}$ and a is a possible value that the random variable A can take. All random variables considered in this paper are assumed to be defined over the real domain $\mathbb{R}$ unless otherwise noted.

3.1 Our Proposed Generative Model

Motivated by cutset Bayesian networks (Rahman et al., 2019), we model the full joint distribution over a set of random variables $\boldsymbol{Z}$ as the product of two parts: a complex local distribution $p(\boldsymbol{X})$ over a small subset of variables $\boldsymbol{X} \subset \boldsymbol{Z}$ and a fully tractable conditional distribution $p(\boldsymbol{Y}|\boldsymbol{X})$ over $\boldsymbol{Y} = \boldsymbol{Z} \backslash \boldsymbol{X}$. Technically, we can use any complex, parametric or non-parametric distribution to represent $p(\boldsymbol{X})$; the only constraint is that the distribution should be easy to sample from (since we will use cutset sampling (Gogate and Dechter, 2005; Bidyuk and Dechter, 2007; Gogate, 2009) for inference). For simplicity, we use a mixture of multivariate Gaussian (MixMG) distribution to model $p(\boldsymbol{X})$.

We propose to use a *templated* Gaussian Bayesian Network (GBN) to model the distribution $p(\boldsymbol{Y}|\boldsymbol{X})$ such that we obtain a multi-variate Gaussian density over $\boldsymbol{Y}$ for each assignment $\boldsymbol{X} = \boldsymbol{x}$. The parameters of the templated GBN are given by a neural network (NN) that takes an assignment $\boldsymbol{x}$ as input and outputs a GBN over $\boldsymbol{Y}$, i.e., $p(\boldsymbol{Y}|\boldsymbol{x}) \sim GBN(\boldsymbol{Y};\theta)$ where $\theta = NN(\boldsymbol{x})$.[1] The NN models non-linear dependence between $\boldsymbol{X}$ and $\boldsymbol{Y}$ and yields a potentially infinite Gaussian mixture model over $\boldsymbol{Z}$.[2]

Note that any GBN can be converted into an equivalent Multivariate Gaussian (MG) distribution with full covariance matrix (Koller and Friedman, 2009). Therefore, the conditional distribution $p(\boldsymbol{Y}|\boldsymbol{X} = \boldsymbol{x})$ is fully tractable, which means our model admits tractable inference when all variables in the local distribution $p(\boldsymbol{X})$ are observed. When $\boldsymbol{X}$ is not fully observed, cutset sampling can be employed to efficiently generate accurate predictions in practice.

In the following sections, we will first introduce the detailed architecture of our model, this includes the architecture of the NN, as well as how we select the conditional variables $\boldsymbol{X}$ from $\boldsymbol{Z}$, and then we will discuss learning and inference algorithms.

3.2 Design of Parameter Generation Neural Network

In this work, we propose a neural network architecture that we dub parameter generation neural networks (PGNNs) that can be used to generate the parameters for other models (specifically GBNs).

The input to the PGNN is an assignment $\boldsymbol{x}$ to all variables in the set $\boldsymbol{X}$ and the output is a set of parameters for a GBN. We assume that we are given the directed acyclic graph G associated with the GBN. Namely we assume that each (conditional) GBN has the same structure but different parameters which are controlled by a neural network. More formally, let $\boldsymbol{Y} = \{Y_1, \ldots, Y_m\}$ and let $\boldsymbol{S}_i \subseteq \boldsymbol{Y}$ be the parents of Y_i in G. Then, the conditional linear Gaussian distribution at each node Y_i is given by

$$p(Y_i|\boldsymbol{S}_i = \boldsymbol{s}_i) \sim N(Y_i; \mu_i = \boldsymbol{w}_i^T \boldsymbol{s}_i + b_i, \sigma_i^2),$$

where the mean μ_i of the normal distribution over Y_i is a linear function of its parent variables $\boldsymbol{S}_i$ controlled by the weight vector $\boldsymbol{w}_i$ as well as a bias term b_i and the conditional standard deviation is denoted as σ_i, which must be greater than zero. Thus, given $\boldsymbol{x}$, the PGNN outputs a triple $(\boldsymbol{w}_i, b_i, \sigma_i)$ for each variable Y_i. Our proposed PGNN architecture is shown in Fig. 1. It is composed of a series of building blocks that are used to extract a shared feature vector stage by stage and a collection of headers that are used to compute the parameters based on the shared feature vector. The architecture consists of a basic building block (BBlock) that is used to extract non-linear features. It consists of three layers: (1) a (fully connected) linear layer that produces linear features, (2) a one-dimension batch normal layer, and (3) finally a ReLU layer for non-linearity. The Header block consists of a BBlock used to refine the shared features and a linear layer that is used to compute the parameters. Given a input vector of size n, we first stack $s+1$ building blocks to extract a feature vector of size k that is shared by all headers. The first building block creates a large and expressive feature vector of size $k \cdot r^s$, and the following s building blocks compress the large vector stage by stage until we reach a feature vector with size k. The size of the feature vector is reduced by a factor of r each time it passes through a building block.

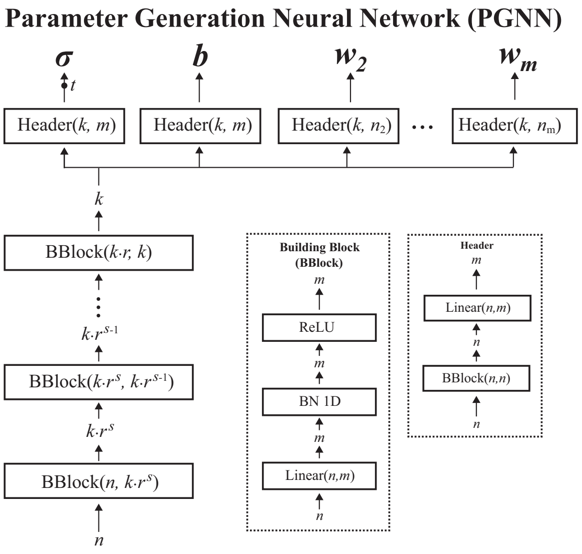

Figure 1: Structure of Parameter Generation Neural Network (PGNN), n is the size of input vector, k is the size of extracted shared feature vector, $r > 1$ is the compress ratio and s controls number of stages.

Once the shared feature vector is built, the individual headers (output layers) generate the parameters for the underlying model. Note that we do not use a single header for all parameters to encourage diversity among the different types of generated parameters. Similarly, the model does not include a header for each parameter as, due to the large number of parameters, the computation complexity can be prohibitive and this could lead to overfitting. The compromise, which we adopt here, is to use multiple smaller headers, one for each group of parameters.

Specifically, we first create a parameter group $\{\sigma_i | i =$

[1] Any regression method can be used instead of the NN.

[2] If we use linear regression, our model yields a multivariate Gaussian density over $\boldsymbol{Z}$. However, when we use nonlinear regression, we obtain a potentially infinite mixture of Gaussians.

$1, \dots, m\}$ and use a single header to predict all standard deviations. Then, we do the same for all bias terms $\{b_i | i = 1, \dots, m\}$. And finally, we construct one header for each of the weight vectors $\boldsymbol{w}_i$ with size equal to $n_i = |\boldsymbol{S}_i|$. When Y_i has no parents, we simply remove the corresponding header. Note that the output of the $\boldsymbol{\sigma}$ header is clipped to a predefined threshold $t > 0$ in order to guarantee validity of $\boldsymbol{\sigma}$ as well as to ensure numerical stability.

3.3 Variable Selection

Clearly, the performance of our model is highly dependent on the selection of the conditional variables, $\boldsymbol{X}$, and the number of mixture components used in $p(\boldsymbol{X})$. In order to circumvent overfitting and underfitting, we use a validation set to determine $|\boldsymbol{X}|$ (size of $\boldsymbol{X}$) and the number of mixture components (namely, we treat them as hyper-parameters).

Once the size of $\boldsymbol{X}$, denoted by k is determined, we employ the following heuristic method to select the k conditional variables. We first use Principal Components Analysis (PCA) to analyze the amount of variance explained by each variable and use it to rank variables from highest to lowest. Intuitively, the more variance a feature explains, the more informative it is. Therefore, we simply choose the first k variables to be the conditional variables $\boldsymbol{X}$. We also experimented with other variable selection heuristics that rank variables based on their pairwise mutual information score, variance, etc. We found experimentally that on average, the PCA based method is superior to other methods. All experimental results reported in this paper use the PCA based method for selecting $\boldsymbol{X}$.

3.4 Inference

The main prediction task in our model is computing the most likely assignment to a subset of (query) variables given observations on another disjoint subset of variables (evidence). Next, we describe algorithms for solving this prediction task, which is called marginal max-a-posteriori (MMAP) inference. A special case of MMAP inference is MAP inference in which all unobserved variables are query variables. This (MMAP) prediction task frequently arises in structured prediction where the model has hidden variables whose predictions are not required at test time but data over these hidden variables is available at training time.

Formally, given evidence $\boldsymbol{Z}_e = \boldsymbol{z}_e$, query variables $\boldsymbol{Z}_q$ and missing variables $\boldsymbol{Z}_m$, MMAP inference aims to find the best assignment $\boldsymbol{z}_q^*$ such that $p(\boldsymbol{z}_q|\boldsymbol{z}_e) = \int_{\boldsymbol{z}_m} p(\boldsymbol{z}_q, \boldsymbol{z}_m|\boldsymbol{z}_e) d\boldsymbol{z}_m$ is maximized. Because our model partitions the variables into two sets $\boldsymbol{X}$ and $\boldsymbol{Y}$, we denote the query, missing, and evidence variables from the local distribution as $\boldsymbol{X}_q$, $\boldsymbol{X}_m$, and $\boldsymbol{X}_e$ and those in the conditional part as $\boldsymbol{Y}_q$, $\boldsymbol{Y}_m$, and $\boldsymbol{Y}_e$ respectively. The inference task can be formulated as finding $\boldsymbol{x}_q^*$ and $\boldsymbol{y}_q^*$ such that

$$
\begin{aligned}
\boldsymbol{x}_q^*, \boldsymbol{y}_q^* &\in \operatorname*{argmax}_{\boldsymbol{x}_q, \boldsymbol{y}_q} \int_{\boldsymbol{x}_m} \int_{\boldsymbol{y}_m} p(\boldsymbol{x}_{q,m}, \boldsymbol{y}_{q,m} | \boldsymbol{x}_e, \boldsymbol{y}_e) \mathrm{d}\boldsymbol{y}_m \mathrm{d}\boldsymbol{x}_m \\
&= \operatorname*{argmax}_{\boldsymbol{x}_q, \boldsymbol{y}_q} \int_{\boldsymbol{x}_m} \int_{\boldsymbol{y}_m} p(\boldsymbol{x}_{q,m,e}, \boldsymbol{y}_{q,m,e}) \mathrm{d}\boldsymbol{y}_m \mathrm{d}\boldsymbol{x}_m \\
&= \operatorname*{argmax}_{\boldsymbol{x}_q, \boldsymbol{y}_q} \int_{\boldsymbol{x}_m} \int_{\boldsymbol{y}_m} p_{\boldsymbol{X}}(\boldsymbol{x}_{q,m,e}) p_{\boldsymbol{Y}}(\boldsymbol{y}_{q,m,e} | \boldsymbol{x}_{q,m,e}) \mathrm{d}\boldsymbol{y}_m \mathrm{d}\boldsymbol{x}_m \\
&= \operatorname*{argmax}_{\boldsymbol{x}_q, \boldsymbol{y}_q} \int_{\boldsymbol{x}_m} p_{\boldsymbol{X}}(\boldsymbol{x}_{q,m,e}) p'_{\boldsymbol{Y}}(\boldsymbol{y}_{q,e} | \boldsymbol{x}_{q,m,e}) \mathrm{d}\boldsymbol{x}_m,
\end{aligned}
$$

where $p'_{\boldsymbol{Y}}$ is the marginal distribution obtained by integrating out $\boldsymbol{Y}_m$ from $p_{\boldsymbol{Y}}$ and $\boldsymbol{x}_{q,m}$ denotes the composition of $\boldsymbol{x}_q$ and $\boldsymbol{x}_m$. The difficulty of the above MMAP problem depends on whether the sets $\boldsymbol{X}_q$ and $\boldsymbol{X}_m$ are empty or not.

Case 1: All variables in $\boldsymbol{X}$ are observed. This means $\boldsymbol{X}_q, \boldsymbol{X}_m = \varnothing$, and the distribution over the remaining variables $\boldsymbol{Y}$ is a GBN with parameters $\theta = NN(\boldsymbol{x})$. In this case, we can do *exact* MMAP inference by converting the GBN into a multivariate Gaussian distribution.

Case 2: $\boldsymbol{X}_q \neq \varnothing$ while $\boldsymbol{X}_m = \varnothing$. In this case, since the joint distribution is not tractable, strategy similar to cutset sampling can be employed to efficiently answer the above query. Compared to standard sampling techniques that sample all the variables, cutset sampling only samples a subset of variables $\boldsymbol{X}$ and does exact inference over the remaining variables $\boldsymbol{Y}$, which makes the inference more accurate (Gogate and Dechter, 2005; Bidyuk and Dechter, 2007). In our case, we first generate n samples $\boldsymbol{x}_q^{(1)}, \dots, \boldsymbol{x}_q^{(n)}$ for $\boldsymbol{X}_q$ from the distribution $p_{\boldsymbol{X}}$ given evidence $\boldsymbol{x}_e$. After that, for each $\boldsymbol{x}_q^{(i)}$, we do exact MMAP inference on $p'_{\boldsymbol{Y}}$ to find the best assignment $\boldsymbol{y}_q^{(i)}$ for $\boldsymbol{Y}_q$ such that $p'_{\boldsymbol{Y}}(\boldsymbol{y}_q^{(i)}, \boldsymbol{y}_e | \boldsymbol{x}_q^{(i)}, \boldsymbol{x}_e)$ is maximized. Finally, we evaluate the joint density for each pair $(\boldsymbol{x}_q^{(i)}, \boldsymbol{y}_q^{(i)})$ along with the evidence and treat the one with maximum density as the MMAP result. Note that the size of $\boldsymbol{X}$ is usually small in our case. Therefore, the number of variables being sampled is small, which enables the cutset sampling inference to generate high quality approximations in practice.

Case 3: There are missing variables in $\boldsymbol{X}$, which means $\boldsymbol{X}_m \neq \varnothing$. In this case, we treat those missing variables as part of the query variables and use the cutset sampling method described in Case 2 to find best assignment $\boldsymbol{x}_{q,m}^*$ for both $\boldsymbol{X}_q$ and $\boldsymbol{X}_m$. Then we simply discard $\boldsymbol{x}_m^*$ and return $\boldsymbol{x}_q^*$ as the MMAP result for $\boldsymbol{X}_q$. This is a popular method in practice used for approximating MMAP by using the partial assignment from the MAP results (Liu and Ihler, 2013; Poon and Domingos, 2011).

3.5 Learning

Given a dataset D defined over variables $\boldsymbol{Z}$, our learning algorithm works as follows. First, we use our variable selection method to select a small subset of variables $\boldsymbol{X} \subset \boldsymbol{Z}$ as the conditional variables. Then, we learn a mixture of multivariate Gaussian (MixMG) over $\boldsymbol{X}$ using the expectation maximization algorithm, where the number of components is automatically determined by tuning on the validation set. After that, we build an undirected complete graph G over all variables in $\boldsymbol{Y} = \boldsymbol{Z} \backslash \boldsymbol{X}$ and remove the edge between Y_i and Y_j if the Pearson correlation between them is less than 0.05. We use a random ordering of $\boldsymbol{Y}$ variables to convert G into a directed acyclic graph, which gives us the structure of template GBN. Finally, we train a PGNN for generating the parameters of the GBN using the *negative log-likelihood* of D as the loss function.

4 EXPERIMENTS

We conducted two different sets of experiments to systematically evaluate our model's performance on prediction tasks. In the first set of the experiments, we evaluate our model's MAP prediction performance through various image completion tasks on the MNIST dataset (LeCun et al., 1998). For the second set of experiments, we perform a comprehensive simulated prediction task on ten real UCI datasets (Dua and Graff, 2017) to evaluate our model's MMAP prediction performance in the case of missing data.

Three competitors are considered in our experiments.

Multivariate Gaussians (MGs) with full covariance matrix. This is considered as a baseline model.

Real-valued Neural Autoregressive Density Estimators (RNADEs) (Uria et al., 2013, 2016) are equivalent to a fully connected Bayesian Network where each of the conditional distributions that define the model are given by a mixture of multivariate Gaussians whose parameters are controlled by neural networks. We developed an approximate MMAP inference algorithm for RNADE since there are no inference algorithms described in the original paper. Specifically, we first employ likelihood weighting to generate N samples for both the query and the missing variables given the evidence. After that, we choose the sample that achieves highest likelihood as the initial point, and perform gradient ascent to further optimize the assignment for a fixed number of iterations. Finally, we return the iteration with the highest log-likelihood. Note that the gradient is approximated using numerical methods since a closed form gradient is not available for RNADE. In addition, backtracking line search is used to determine an appropriate step size in each iteration. For the tuning of hyperparameters, we tune the number of components, $\{2, 5, 10, 20\}$; the weight-decay, $\{0.1, 0.01, 0.001, 0\}$; and the learning rate $\{0.1, 0.05, 0.025, 0.0125\}$. The remaining hyperparameters, where possible, are set following Uria et al. (2013), left at the default value determined by existing code, or set to a value that is equivalent to our method if we have the same hyperparameter. For example, both our model and RNADE use the same number of pre-trainings.

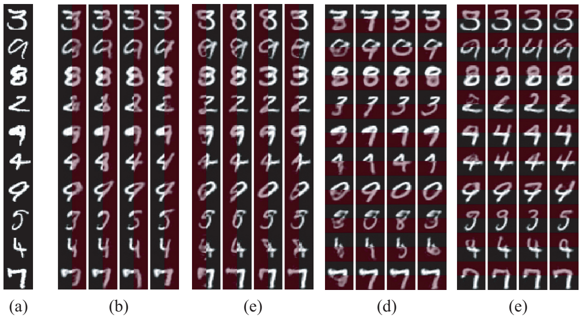

Figure 2: Image completion results where (a) is the ground truth and (b-e) is the completion results when right, left, bottom, top are missing (marked in red), respectively. The order in group (b-e) are organized as MG, RNADE, SPN, OURS from left to right.

Sum-Product Networks (SPNs) (Poon and Domingos, 2011; Molina et al., 2018) admit tractable MAP inference (under certain constraints). However for MMAP inference, SPNs are not tractable, but MMAP can be approximated by first approximately marginalizing out the missing variables and then applying MAP inference over the marginalized SPN. We implement SPNs using existing libraries (Molina, 2019; Molina et al., 2019; Lorenzo, 2020), and tune three different hyperparameters: the instance threshold for creating leaf nodes, $\{50, 100, 150, 200, 250, 300, 400, 500\}$; the row splitting method, $\{kmeans, gmm, rdc\}$; and the column splitting method, $\{rdc, gvs\}$. Note that we restrict the leaf nodes in SPNs to univariate Gaussian distributions and thus we do not need to tune the feature threshold for creating product nodes.

PGNNs (our model) is trained using the Adam optimizer from pyTorch (Paszke et al., 2019) with fixed batch size 100, and weight decay $1e^{-3}$ for 150 epochs. For each of the experiments, the number of stages $s = 3$, the reduction factor $r = 2.5$, and the standard deviation threshold $t = \sqrt{5e^{-3}}$ are fixed. In addition,

Table 1: Prediction RMSE of ten UCI datasets under nine query and missing settings. Our model achieves the best average prediction error over the ten dataset for all nine scenarios.

Models	**Datasets**										**Average**
	airquality	energy	hepmass	miniboone	onlinenews	parkinson	sdd	superconduct	wec sydney	cropmapping	
Query: 10% Missing: 10%											
MG	0.2407	0.2810	**0.8146**	0.4165	0.3875	0.3495	0.4744	0.1529	0.5586	**0.2237**	0.4084
RNADE	0.5851	0.2639	**0.7803**	0.4117	0.6703	0.3720	0.4616	0.2660	0.6451	0.9620	0.4951
SPN	0.1527	0.1926	0.9040	0.4980	0.4518	0.3899	**0.3469**	0.1933	0.5695	0.3245	0.4110
OURS	**0.1451**	**0.1316**	**0.8011**	**0.3854**	**0.3296**	**0.2432**	0.5088	**0.0723**	**0.4368**	0.2442	**0.3393**
Query: 10% Missing: 20%											
MG	0.2879	0.3251	0.7875	0.4598	**0.4963**	0.4126	0.4508	0.1929	0.5961	**0.2301**	0.4455
RNADE	0.5537	0.3354	**0.7351**	0.4414	0.7595	0.4559	0.4657	0.3388	0.6439	0.9841	0.5255
SPN	0.1646	**0.1921**	0.8787	0.5209	0.5162	0.4420	**0.3245**	0.1982	0.5389	0.3399	0.4196
OURS	**0.1567**	**0.1972**	0.7823	**0.3848**	**0.4783**	**0.3565**	0.4797	**0.0866**	**0.4554**	0.2512	**0.3753**
Query: 10% Missing: 30%											
MG	0.3677	0.3581	0.7840	0.4744	0.5286	0.3842	0.4977	0.1975	0.6175	**0.2570**	0.4678
RNADE	0.6055	0.3312	**0.7370**	0.5020	0.8479	0.4356	0.5346	0.3061	0.6782	1.0154	0.5531
SPN	**0.2201**	**0.1960**	0.8436	0.5530	0.5870	0.4350	**0.3996**	0.1933	0.5758	0.3351	0.4448
OURS	**0.2313**	**0.1889**	0.7877	**0.3957**	**0.4632**	**0.3122**	0.4884	**0.0967**	**0.5097**	0.2906	**0.3860**
Query: 20% Missing: 10%											
MG	0.3192	0.3723	**0.8573**	0.4655	0.4895	0.4237	0.5266	0.1940	0.6087	**0.2675**	0.4730
RNADE	0.6665	0.3674	**0.8497**	0.4637	0.7921	0.4782	0.5255	0.3310	0.6885	1.0217	0.5736
SPN	**0.1920**	**0.2047**	0.9458	0.5247	0.5649	0.4389	**0.4072**	0.2075	0.5697	0.3472	0.4506
OURS	**0.1898**	**0.2081**	**0.8466**	**0.4349**	**0.4522**	**0.3824**	0.5232	**0.0997**	**0.4769**	0.2915	**0.4015**
Query: 20% Missing: 20%											
MG	0.3400	0.4359	**0.8744**	0.5023	**0.5878**	0.4219	0.5089	0.2091	0.6526	**0.2525**	0.5037
RNADE	0.7200	0.4169	**0.8957**	0.5272	0.8427	0.4200	0.5568	0.3267	0.6885	1.0757	0.5994
SPN	**0.2535**	**0.2082**	0.9862	0.5493	0.6190	0.4591	**0.4224**	0.1989	0.5844	0.3257	0.4757
OURS	**0.2556**	0.2727	**0.8737**	**0.4287**	**0.5770**	**0.3823**	0.5130	**0.0976**	**0.5249**	0.2872	**0.4362**
Query: 20% Missing: 30%											
MG	0.3879	0.4355	**0.8285**	0.5154	**0.6271**	0.4111	0.5405	0.2425	0.6754	**0.2579**	0.5182
RNADE	0.7509	0.4334	0.8704	0.5333	0.9257	0.4496	0.5876	0.3404	0.7156	1.0375	0.6230
SPN	**0.2655**	**0.2119**	0.9483	0.5526	0.7053	0.4499	**0.4698**	0.2056	0.6112	0.3330	0.4911
OURS	0.2811	0.2749	**0.8234**	**0.4604**	**0.6003**	**0.3835**	0.5525	**0.1210**	**0.5328**	0.2961	**0.4478**
Query: 30% Missing: 10%											
MG	0.361	0.418	**0.881**	0.487	0.529	0.458	0.544	0.215	0.664	**0.281**	0.506
RNADE	0.863	0.402	0.925	0.513	0.897	0.516	0.619	0.330	0.717	1.060	0.643
SPN	**0.272**	**0.230**	1.002	0.552	0.581	0.491	**0.495**	0.203	0.602	0.342	0.492
OURS	0.356	0.247	**0.878**	**0.439**	**0.498**	**0.418**	0.572	**0.096**	**0.527**	0.313	**0.448**
Query: 30% Missing: 20%											
MG	0.4232	0.4234	**0.8978**	0.5293	**0.6120**	0.4942	0.5623	0.2564	0.7001	**0.2793**	0.5443
RNADE	0.8547	0.4310	0.9816	0.5377	0.9821	0.5445	0.6073	0.3621	0.7042	1.0807	0.6673
SPN	**0.2965**	**0.2415**	1.0288	0.5578	0.7108	0.5289	**0.5110**	0.2080	0.5941	0.3372	0.5197
OURS	0.3472	0.2937	**0.9009**	**0.4633**	**0.5842**	**0.4606**	0.5714	**0.1359**	**0.5420**	0.3079	**0.4777**
Query: 30% Missing: 30%											
MG	0.4928	0.4390	**0.8857**	0.5758	**0.7050**	0.4864	0.5776	0.2893	0.7503	**0.3107**	0.5780
RNADE	0.8830	0.4834	0.9823	0.6079	0.9878	0.5501	0.9062	0.3965	0.7612	1.1024	0.7287
SPN	**0.4184**	**0.2384**	1.0419	0.5735	0.8257	0.5574	**0.5105**	0.2125	0.6421	0.3590	0.5578
OURS	0.5034	0.3495	**0.9014**	**0.5059**	**0.7082**	**0.4215**	0.5889	**0.1548**	**0.5938**	0.3440	**0.5253**

Table 2: Loglikelihood and RMSE of MAP assignment for different image completion tasks.

Models		MG	RNADE	SPN	OURS
MAP LL	Right	**49.003**	-59.100	-61.856	-13.108
	Left	**49.552**	-58.061	-60.997	-11.381
	Bottom	**67.826**	-56.063	-62.004	-0.548
	Top	**68.966**	-56.890	-60.271	-0.796
MAP RMSE	Right	0.1698	**0.1572**	**0.1603**	**0.161**
	Left	0.1664	0.1549	0.154	**0.1436**
	Bottom	0.1738	0.1699	0.1719	**0.1573**
	Top	0.1693	0.1659	**0.164**	**0.158**

Table 3: Test set LL of image completion tasks.

Test LL	MG	RNADE	SPN	OURS
$P(\boldsymbol{X}_l, \boldsymbol{X}_r)$	**37.203**	-73.419	-77.708	-25.331
$P(\boldsymbol{X}_t, \boldsymbol{X}_b)$	**56.793**	-71.771	-78.909	-12.727

the learning rate update is scheduled using a linear warm up of 5 epochs starting from 1% of the maximum learning rate, followed by cosine decay. As for the size of the shared feature vector k, it is highly dependent on the data dimension; we make it a hyper-parameter and use a validation set to determine the best value from a predefined candidate set. Note that the dimensions of different datasets can vary greatly, and the candidate set for parameter k might be too large, incurring significant tuning overhead. Therefore, we decided to make the candidate set for parameter k adaptive to the given data by tuning the feature ratio fr and then using it to determine the parameter k. Specifically, we first apply PCA to the dataset and then find the minimum number of components k such that the total variance explained by those components is larger than a given percentage fr (called feature ratio). At last, following Uria et al. (2013, 2016), we pretrain 4 times for 15 epochs each and select the model that achieves best loss for subsequent training. For hyperparameter tuning, we considered base maximum learning rate, $\{0.025, 0.01, 0.004\}$; maximum number of conditional variables, $\{10\%, 20\%, 30\%, 40\%\}$; and the feature ratio $\{0.9, 0.95, 0.97, 0.99\}$.

We compare our model against the above three competitors by evaluating both the test set loglikelihood (LL) as well as the root mean square error (RMSE) of the MAP/MMAP inference results. Note that most existing research on *continuous* domains only evaluate their model using the loglikelihood, e.g., (Molina et al., 2018; Uria et al., 2013, 2016), with some exceptions (Strauss and Oliva, 2021) that also present prediction/imputation results evaluated by RMSE-like metrics. Our results indicate that loglikelihood can sometimes be misleading as it is not bounded in continuous domains. As a result, it may not be able to accurately reflect the model's true performance. Therefore, we present results with respect to both loglikelihood and prediction RMSE.

Table 4: One-to-One comparison of win/tie/lose achieved by one model (row) over the other (column) on prediction tasks with missing variables.

	MG	RNADE	SPN	OURS
MG	-	65/21/4	44/6/40	12/22/56
RNADE	4/21/65	-	14/10/66	3/5/82
SPN	40/6/44	66/10/14	-	19/6/65
OURS	56/22/12	82/5/3	65/6/19	-

For each dataset, we randomly selected 200 instances for both validation and testing. The remaining instances are used as the training split. All models (SPN, RNADE, OURS) are tuned based on the best RMSE achieved on the validation set as we found that tuning with respect to loglikelihood tends to overfit the training data and often leads to poor MAP/MMAP inference results. In addition, for sample based approximate inference schemes, the sample size is based on the number of query and missing variables during inference, with a minimum number of 200 samples.

All experiments were conducted on a workstation with a 16-core Intel Xeon Gold 6130 CPU and two Quadro P5000 GPUs. Note that running on a GPU is not required for our model as the PGNNs are typically small compared to modern deep neural networks. For a small datasets like UCI Parkinson, CPU only performance is 2x faster than using GPUs. For larger datasets like UCI cropmapping, running our model on GPUs can achieve a 3x to 5x speedup over CPUs. All datasets and codes are publicly available on GitHub[3].

4.1 Image Completion

In this set of experiments, we consider the image completion problem on the MNIST dataset to evaluate our model's MAP prediction performance. Following Molina et al. (2018), we process the original 28×28 MNIST image as two halves: left $\boldsymbol{X}_l$ and right $\boldsymbol{X}_r$, top $\boldsymbol{X}_t$ and bottom $\boldsymbol{X}_b$. We fit an auto-encoder to each half and featurize the half images into a 25 dimensional vector. Then we train our model along with other three competitors to learn the joint distribution $P(\boldsymbol{X}_l, \boldsymbol{X}_r)$ and $P(\boldsymbol{X}_t, \boldsymbol{X}_b)$. Finally, we perform MAP inference on the models to predict one half of the image given the other half as evidence for each test instance.

Table 2 shows the loglikelihood and RMSE of the MAP assignments for the four image completion tasks. We

[3] Github Repository - TractableDE-ContCNet

Table 5: Test set loglikelihood of ten UCI datasets.

Test LL	MG	RNADE	SPN	OURS
airquality	-7.639	4.691	**29.682**	12.743
energy	-13.627	-5.849	**8.657**	2.785
hepmass	-28.348	**-21.773**	-25.801	-26.011
miniboone	-37.182	**-21.01**	-30.026	**-21.421**
onlinenews	-27.195	6.772	64.882	-32.274
parkinson	-10.566	-3.899	-9.706	**-3.313**
sdd	-26.362	6.215	**18.526**	-8.882
superconduct	-4.764	26.918	**176.616**	67.913
wec sydney	-48.627	-29.862	**-20.65**	-24.119
cropmapping	33.762	-116.632	46.459	**97.908**
Average	-17.055	-15.443	**25.864**	6.533

also evaluated the test set loglikelihood (see Table 3). The best results are marked in bold and we consider results within 5% difference as a tie. To better demonstrate the prediction quality of different models, in Figure 2, we show qualitative results for a few instances chosen from the test set such that the prediction among the models differs the most. More visualizations can be found in Appendix.

We have several observations. (1) Our model achieves the best or comparable RMSE as well as relatively high MAP and test set LL in all four cases. In addition, as shown in Figure 2, our model can generate correct and plausible completions in most cases. In cases where our model gives the wrong completions, the images often still look like valid digits. (2) SPNs achieve slightly better RMSE despite its lower MAP and test set loglikelihood scores compared to RNADE. From the visualization results, SPNs are more likely to generate incorrect completions compared to our model, while the prediction quality of RNADE is worse than both ours and SPNs in most cases. (3) MG achieves extremely high test set and MAP loglikelihood. However, the RMSE and the completion it generates is the worst among all the models. This is not surprising as Gaussian distributions are unimodal, but the MNIST dataset is obviously not unimodal (it has ten different classes of images). (4) We observed that loglikelihood-like metrics in continuous domains can be misleading in some cases. In other words, high loglikelihood may not translate into good prediction performance in practice. Recall that the loglikelihood is not bounded in continuous domains. As a result, the test set LL can be heavily biased by some of the test instances (this can be counteracted with (much) larger test sets but this may not be realistic in practice).

4.2 Prediction with Missing Variables

In this experiment, we evaluate our model's MMAP performance through a comprehensive series of simulated prediction tasks on ten publicly available UCI datasets. Following Uria et al. (2013), we preprocess the datasets by eliminating discrete valued features and one of the attributes from every pair of attributes whose Pearson correlation coefficient is greater than 0.98. The number of instances and features for each dataset after pre-processing is included in the supplementary material. All datasets were normalized by subtracting the mean and then dividing by the standard deviation. We consider nine different settings of the query and missing variables where the percentage of query and missing variables is chosen from the set $\{10\%, 20\%, 30\%\}$. For each setting, we randomly assign query and missing variables for *each* of the instances in test set. Table 1 shows the average RMSE and Table 5 reports the test set loglikelihood of our model along with the other competitors. As before, the best results are marked in bold, and we consider results within 5% difference as a tie. In addition, Table 4 presents a one-to-one comparison of win/tie/lose for one model (row) over another (column).

From the above results, we have the following observations.

First, our model has significantly better average RMSE for all nine query and missing settings despite being only the second best model in terms of test set loglikelihood. In addition, we achieve the best or comparable RMSE for over 65% of the cases against all other competitors.

Second, SPNs are the second best model overall; they achieve the best test set loglikelihoods as well as good prediction RMSE in general. However, the predictive performance is not stable across datasets, and it fails to beat the baseline MG for over 45% of the cases as shown in Table 4.

Third, RNADE, somewhat surprisingly, seems to be the worst one among all models tested. For example, it fails to beat the baseline MG in over 70% of the cases. We believe there are two likely reasons for this. First, RNADE is a complicated model and is intractable everywhere (even the gradient is numerically approximated). This might degrade the performance of the approximate MMAP procedures used in this case. Further, the MMAP techniques are quite general, i.e., they are not specifically hand-tailored to RNADE. Also, the number of hyperparameters that need to be tuned in RNADE is extensive. Uria et al. (2013) manually tuned RNADE on a relatively large dataset. In our case, we only tuned a fraction of the possible hyperparameters as otherwise the search would have been computationally expensive. As such, with additional tuning, the performance of RNADE could likely be improved (though the same could be said of more extensive tuning for SPNs and our model as well).

Finally, MG can produce acceptable or even good results on some datasets (cropmapping). This may be due to the fact that MGs admit exact MMAP inference (no approximation), and good practical performance if the data distribution is well-approximated by a Gaussian.

Note that in the MMAP scenario, the RMSE of the MMAP assignment against the true value of the test instance might not be appropriate as we need to find the most probable assignment to only query variables and average across the missing variables under the given evidence. In other words, neither RMSE nor conditional log-likelihood is a perfect measure but because the former is bounded while the latter is not, RMSE is typically more well behaved and is likely to pick the *correct* best performing scheme. Ideally, if a true model is available, the perfect measure is *conditional loglikelihood of the assignment w.r.t. to the true model.* However, because the true model is often not available, RMSE is a reasonable choice for evaluating the model's predictive performance *in the continuous case.*

While popular as an evaluation metric for continuous models (cf. Uria et al. (2013); Molina et al. (2018)), test-set log-likelihood can result in a misleading picture of a model's practical performance when used to compare *models from different families.* To illustrate this, we evaluated the LL and RMSE of SPNs and our model under different hyperparameters on the validation portion of ten UCI datasets. Figure. 3 shows the result on the `superconduct` dataset (visualizations on other datasets can be found in the supplementary material). The general trend in these results is that if we focus on one model (either ours or SPNs), better predictive performance tends to correlate with higher LL. However, when we compare the LL and RMSE across two models, higher test-set LL does not necessarily translate into better RMSE. As such, caution should be exercised when using LL to compare models across different families, especially in limited data settings.

5 DISCUSSION AND CONCLUSION

We described a flexible family of tractable models for solving MMAP/MAP prediction tasks in continuous domains. Experimentally, we verified that our approach outperforms existing approaches such as SPNs and RNADE on a variety of prediction tasks. In particular, we found that (1) although SPNs typically produced models with higher test set log-likelihoods, our model typically resulted in the best performance on the prediction task and (2) the number of hyperparameters required to fit RNADE in practice was computationally prohibitive and/or much more careful hand-tuning may be required, which limits it applicability.

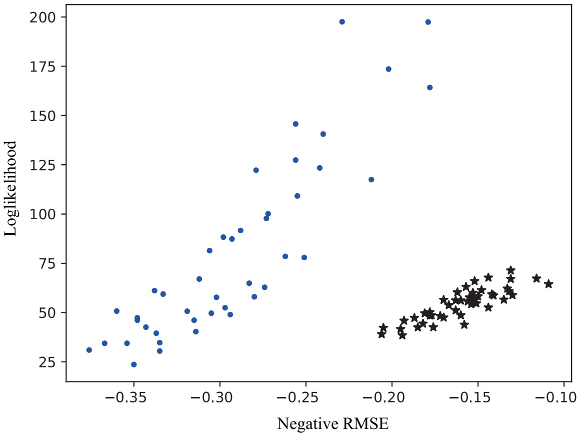

Figure 3: RMSE and loglikelihood achieved on validation dataset (superconduct) for OURS (black stars) and SPN (blue dots) under different hyper parameters.

Compared to other existing tractable models or density estimators, our model can generate more accurate predictions while still achieving a reasonable test set loglikelihood. In particular, our model is ideally suited for applications where a clear delineation exists between query and observed variables. In such cases, we can choose variables that are likely to be observed at query time to be part of the set $\boldsymbol{X}$. This is because our predictions are exact or close to it when the set of observed variables is included in the set $\boldsymbol{X}$.

One of the main limitations of our model is that, it is not tractable when calculating the marginal density of a partial assignment if some of the conditional variables are missing. This issue can be alleviated if we choose variables that have high chance to be observed as conditional variables. Also, the PGNNs used in our model have potential scalability issues as the size of Header blocks is quadratic w.r.t. the number of variables. However, this can be easily solved by projecting the data into low-dimensional space or restricting the number of parents a node can have inside the template GBN. The model is prone to overfitting and in practice we can alleviate this limitation by using simpler models (e.g., linear regression) for some conditional distributions $p(Y_i|\boldsymbol{S}_i, \boldsymbol{x})$ and using learning methods that are robust to minor perturbations (e.g., adversarial learning).

We are also interested in exploring applications of this approach to temporal data or more generally in situations where there may be additional graphical structure that can be exploited. In addition, many more neural network structures are possible for parameter selection, and in different applications, it may be of interest to explore different function characterizations of both local distribution $p(\boldsymbol{X})$ and conditional distribution $p(\boldsymbol{Y}|\boldsymbol{X})$. We leave these extensions for future work.

Acknowledgements

This work was supported in part by the DARPA Explainable Artificial Intelligence (XAI) Program under contract number N66001-17-2-4032, by the DARPA Perceptually-enabled Task Guidance (PTG) Program under contract number HR00112220005 and by the National Science Foundation grant IIS-1652835.

References

B. Bidyuk and R. Dechter. Cutset Sampling for Bayesian Networks. *Journal of Artificial Intelligence Research*, 28:1–48, 2007.

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

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

Dheeru Dua and Casey Graff. UCI machine learning repository, 2017. URL `http://archive.ics.uci.edu/ml`.

Mathieu Germain, Karol Gregor, Iain Murray, and Hugo Larochelle. Made: Masked autoencoder for distribution estimation. In *International Conference on Machine Learning*, pages 881–889. PMLR, 2015.

V. Gogate and R. Dechter. Approximate inference algorithms for hybrid Bayesian networks with discrete constraints. In *Proceedings of the Twenty-First Conference on Uncertainty in Artificial Intelligence*, pages 209–216, 2005.

Vibhav Gogate. *Sampling Algorithms for Probabilistic Graphical Models with Determinism.* PhD thesis, Computer Science, University of California, Irvine, USA, June 2009.

Marco Grzegorczyk. An introduction to Gaussian Bayesian networks. In *Systems Biology in Drug Discovery and Development*, pages 121–147. Springer, 2010.

Reimar Hofmann and Volker Tresp. Discovering structure in continuous variables using Bayesian networks. In D. Touretzky, M. C. Mozer, and M. Hasselmo, editors, *Advances in Neural Information Processing Systems*, volume 8, pages 500–506. MIT Press, 1996.

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

Doga Kisa, Guy Van den Broeck, Arthur Choi, and Adnan Darwiche. Probabilistic sentential decision diagrams. In *Proceedings of the 14th international conference on principles of knowledge representation and reasoning (KR)*, pages 1–10, 2014.

Daphne Koller and Nir Friedman. *Probabilistic graphical models: principles and techniques.* MIT press, 2009.

S. L. Lauritzen. *Graphical Models.* Oxford University Press, 1996.

S. L. Lauritzen and N. Wermuth. Graphical Models for Associations between Variables, some of which are Qualitative and some Quantitative. *The Annals of Statistics*, 17(1):31 – 57, 1989.

Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11): 2278–2324, 1998.

Qiang Liu and Alexander Ihler. Variational algorithms for marginal MAP. *The Journal of Machine Learning Research*, 14(1):3165–3200, 2013.

Loconte Lorenzo. Sum-product networks and normalizing flows for tractable density estimation [mit license]. `https://github.com/loreloc/spnflow`, 2020.

Alejandro Molina. Sum product flow: An easy and extensible library for sum-product networks [apache license v2.0]. `https://github.com/SPFlow/SPFlow`, 2019.

Alejandro Molina, Antonio Vergari, Nicola Di Mauro, Sriraam Natarajan, Floriana Esposito, and Kristian Kersting. Mixed sum-product networks: A deep architecture for hybrid domains. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018.

Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Pranav Subramani, Nicola Di Mauro, Pascal Poupart, and Kristian Kersting. Spflow: An easy and extensible library for deep probabilistic learning using sum-product networks, 2019.

S. Monti and Gregory F. Cooper. *Learning Hybrid Bayesian Networks from Data*, pages 521–540. Springer Netherlands, Dordrecht, 1998.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. *arXiv preprint arXiv:1705.07057*, 2017.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Kopf, Edward Yang, Zachary DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 8024–8035. Curran Associates, Inc., 2019.

Judea Pearl. *Probabilistic Reasoning in Intelligent Systems: Networks of Plausible Inference*. Morgan Kaufmann, San Francisco, CA, 1988.

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

Tahrima Rahman and Vibhav Gogate. Learning ensembles of cutset networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 30, 2016.

Tahrima Rahman, Prasanna Kothalkar, and Vibhav Gogate. Cutset networks: A simple, tractable, and scalable approach for improving the accuracy of chow-liu trees. In *Joint European conference on machine learning and knowledge discovery in databases*, pages 630–645. Springer, 2014.

Tahrima Rahman, Shasha Jin, and Vibhav Gogate. Cutset Bayesian networks: A new representation for learning rao-blackwellised graphical models. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence*, 2019.

Chiradeep Roy, Tahrima Rahman, Hailiang Dong, Nicholas Ruozzi, and Vibhav Gogate. Dynamic cutset networks. In *International Conference on Artificial Intelligence and Statistics*, pages 3106–3114. PMLR, 2021.

Xiaoting Shao, Alejandro Molina, Antonio Vergari, Karl Stelzner, Robert Peharz, Thomas Liebig, and Kristian Kersting. Conditional sum-product networks: Imposing structure on deep probabilistic architectures. In *International Conference on Probabilistic Graphical Models*, pages 401–412. PMLR, 2020.

Ryan R Strauss and Junier B Oliva. Arbitrary conditional distributions with energy. *arXiv preprint arXiv:2102.04426*, 2021.

Nils Thoma, Zhongjie Yu, Fabrizio Ventola, and Kristian Kersting. Recowns: Probabilistic circuits for trustworthy time series forecasting. *arXiv preprint arXiv:2106.04148*, 2021.

Benigno Uria, Iain Murray, and Hugo Larochelle. Rnade: the real-valued neural autoregressive density-estimator. In *Proceedings of the 26th International Conference on Neural Information Processing Systems (NeurIPS)*, pages 2175–2183, 2013.

Benigno Uria, Marc-Alexandre Côté, Karol Gregor, Iain Murray, and Hugo Larochelle. Neural autoregressive distribution estimation. *The Journal of Machine Learning Research*, 17(1):7184–7220, 2016.

Supplementary Material:
Conditionally Tractable Density Estimation using Neural Networks

A UCI Dataset Information

We show the number of instances and features for each UCI dataset after pre-processing in Table. 6.

Table 6: Number of instances and features of ten UCI datasets (after preprocessing).

Name	#instance	#feature
airquality	9357	12
cropmapping	50000	117
energy	19735	24
hepmass	150000	21
miniboone	36488	43
onlinenews	39644	32
parkinson	5875	15
sdd	58509	29
superconduct	21263	68
wec-sydney	72000	49

B Additional Experimental Results

B.1 Image Completion Visualizations

We present extra visualizations (50 in total) of the image completion results of different models on MNIST dataset. Details can be found in Figure. 4.

B.2 Hyperparameter Space Visualization

To better illustrate a model's predictive performance respect to its loglikelihoods in the continuous case, we evaluated the loglikelihood and RMSE on the validation portion of ten UCI datasets for SPN and our model under different hyper parameters. The result for `superconduct` dataset is shown in Figure. 3, where the results for other nine datasets in shown in Figure. 5.

From these results, it is easy to see that there is *no clear relationship between the loglikelihood and RMSE* across two different models. In other words, we cannot determine the predictive performance of different models by just looking at their loglikelihoods. In addition, even for a single model, the prediction performance can be quite different even though the loglikelihoods are quite close, and vice versa (see results of OURS on `sdd` dataset).

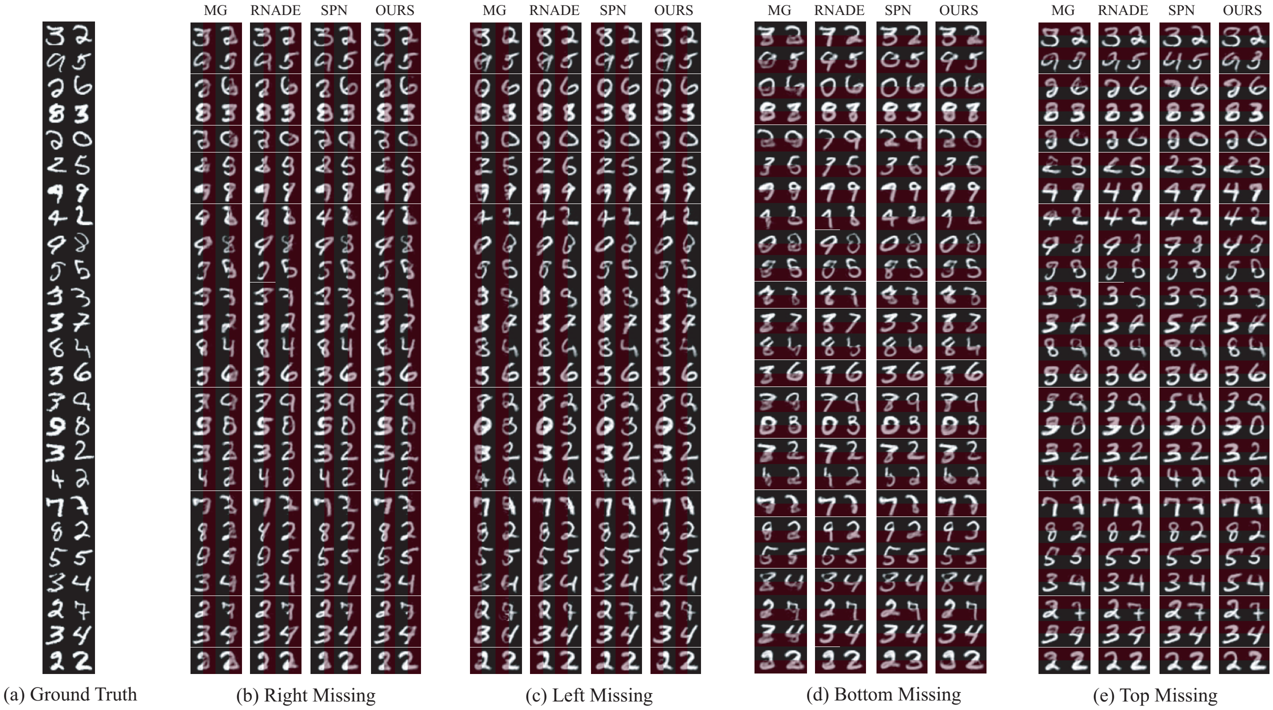

Figure 4: Image completion results where (a) is the ground truth and (b-e) is the completion results when right, left, bottom, top are missing (marked in red), respectively.

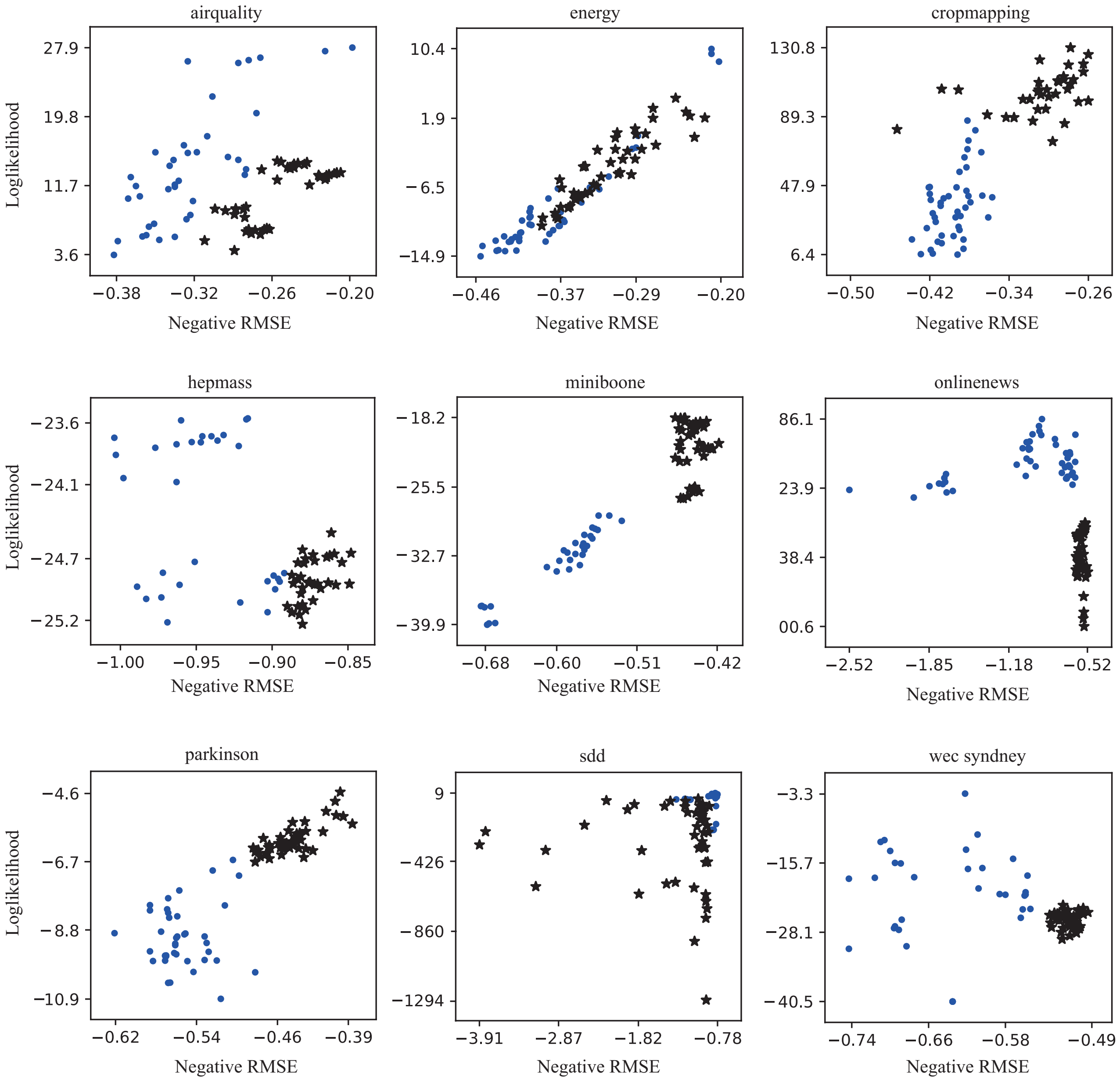

Figure 5: RMSE and loglikelihood achieved on validation portion of 9 UCI datasets for OURS (black stars) and SPN (blue dots) under different hyper parameters.