Whittle Networks: A Deep Likelihood Model for Time Series

Zhongjie Yu [1] **Fabrizio Ventola** [1] **Kristian Kersting** [1 2]

Abstract

While probabilistic circuits have been extensively explored for tabular data, less attention has been paid to time series. Here, the goal is to estimate joint densities among the entire time series and, in turn, determining, for instance, conditional independence relations between them. To this end, we propose the first probabilistic circuits (PCs) approach for modeling the joint distribution of multivariate time series, called Whittle sum-product networks (WSPNs). WSPNs leverage the Whittle approximation, casting the likelihood in the frequency domain, and place a complex-valued sum-product network, the most prominent PC, over the frequencies. The conditional independence relations among the time series can then be determined efficiently in the spectral domain. Moreover, WSPNs can naturally be placed into the deep neural learning stack for time series, resulting in Whittle Networks, opening the likelihood toolbox for training deep neural models and inspecting their behaviour. Our experiments show that Whittle Networks can indeed capture complex dependencies between time series and provide a useful measure of uncertainty for neural networks.

1. Introduction

Probabilistic graphical models specify joint densities compactly using conditional independencies between random variables (RVs). When faced with time series, dynamic Bayesian networks are commonly employed. In many applications, however, one instead often aims to infer the conditional independence relations between time series themselves, accounting for interactions at all possible lags, leading to time series graphical models (TGMs) (Tank et al., 2015). Consider, e.g., the stock price changes of industrial sectors shown in Fig. 1 (Left). The drop in late 2018 illustrates a critical need for new and fundamental understandings of the structure and dynamics of economic networks (Schweitzer et al., 2009).

Arguably, Dahlhaus (2000) introduced the first (undirected) graphical model for stationary time series. Specifically, for jointly Gaussian stationary time series, one transforms the series to the frequency domain and estimates a Gaussian graphical model in the resulting spectral representation. The conditional independencies between time series are encoded by zeros in the inverse spectral density matrix. Bach & Jordan (2004) leveraged the Whittle approximation (Whittle, 1953), casting the likelihood in the frequency domain, and Tank et al. (2015) proposed a Bayesian extension, making use of hyper complex inverse Wishart distribution priors. Unfortunately, using graphical models for time series modeling has a number of important limitations. First, inference is exponential in the worst case. Second, the sample size required for accurate learning is worst-case exponential in scope size, i.e. the subset of variables of each potential. Third, learning requires inference as a subroutine, i.e. it can take exponential time even with fixed scopes.

To overcome these limitations of TGMs and inspired by the successes of deep probabilistic models for univariate time series (Trapp et al., 2020; Melibari et al., 2016), we introduce the first probabilistic circuit for modeling the joint distribution of multivariate time series, called Whittle sum-product network (WSPN). It also leverages the Whittle approximation but places a sum-product network (SPN) (Poon & Domingos, 2011) over the frequencies. Using SPNs in the frequency domain, however, requires different decomposition and conditioning operations for SPNs tailored towards complex-valued RVs—our main technical contributions. The conditional independence relations among the time series, even in a directed fashion, can then be determined efficiently in the spectral domain, see Fig. 1 (Middle).

While WSPNs feature efficient inference, learning their structure—the structure of complex-valued SPNs—can still be tedious and may not scale well to large number of RVs. Therefore, we propose to "go down the deep neural road" one step further by generating unspecialized random structures for the SPNs (Peharz et al., 2020b; Ventola et al.,

[1]Department of Computer Science, TU Darmstadt, Darmstadt, Germany [2]Centre for Cognitive Science, TU Darmstadt, and Hessian Center for AI (hessian.AI). Correspondence to: Zhongjie Yu <yu@cs.tu-darmstadt.de>.

Proceedings of the 38^{th} International Conference on Machine Learning, PMLR 139, 2021.

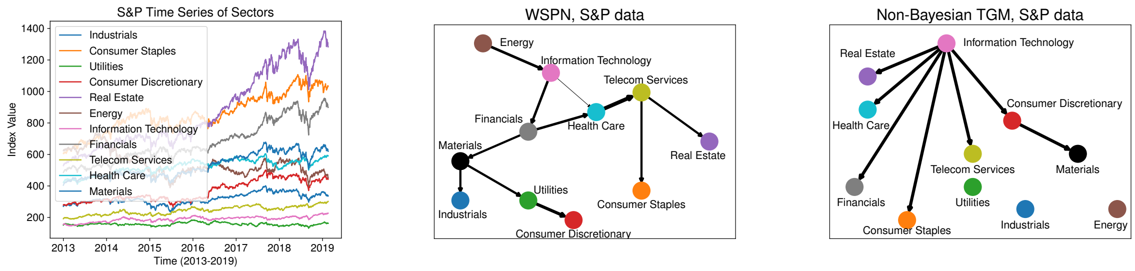

Figure 1. Illustration of a multivariate time series and the discovered independence structures. (Left) Time series of 11 sectors from Standard & Poor's index. (Middle) Conditional independence structure among the sectors discovered by Whittle sum-product network. The thickness of the edge indicates the increase of Whittle likelihood by adding this edge. (Right) Conditional independence structure from Non-Bayesian TGM. (Best viewed in color.)

2020; Stelzner et al., 2019; Kossen et al., 2020), scalable to millions of parameters and trainable in an end-to-end fashion, even together with deep neural networks (NNs) for time series. This results in Whittle Networks, and it opens the likelihood toolbox for training deep neural models for time series and for inspecting their behaviour based on probabilistic grounds. Our experimental results on stock market data, synthetic time series, MNIST, and hyperspectral images demonstrate that Whittle Networks can indeed capture complex dependencies between time series and provide a useful measure of uncertainty for neural networks. To summarize, we make the following contributions:

- The first probabilistic circuit for modeling the joint distribution of multivariate time series, called Whittle sum-product networks (WSPNs), by introducing complex-valued SPNs.
- Using WSPNs, we propose deep likelihood functions for training deep neural networks for time series in an end-to-end fashion, called Whittle Networks. We illustrate this by introducing Whittle Autoencoder, a novel probabilistic autoencoder for time series.

To this end, we proceed as follows. We start off by introducing WSPNs, reviewing more related work on the fly. Then, we show how to read off conditional independencies from them and how to interface them with deep neural networks, resulting in Whittle Networks. Before concluding, we present our empirical evaluation[1].

2. Whittle SPNs: SPNs for Time Series

Neural networks have been widely used for time series processing. For instance, multilayer perceptron (MLP) can work as an autoencoder (AE) for univariate time series modeling (Koskela et al., 1996). Similarly, Convolutional Neural Networks (CNN) have also been used for time series modeling (LeCun & Bengio, 1995). Moreover, Recurrent Neural Networks (Connor et al., 1994) and in particular Long Short-Term Memory (LSTM) Neural Networks (Gers et al., 2000) have been extensively used for neural modeling of time series. However, the above neural models can not provide a natural probability measure of the outputs.

On the other hand, there are deep generative models that use well-defined likelihood functions. State space model and deep neural networks are combined in Rangapuram et al. (2018). Similarly, state space model with conditional probability is investigated also for reinforcement learning in Buesing et al. (2018). Kalchbrenner et al. (2017) encode time, space, and color structure into a dependency chain to model video sequences. However, these works can answer to a considerably limited set of queries because they are restricted only to forecasting and do not model the joint distribution of the complete time series. A generative approach has been proposed in Krishnan et al. (2017) but it offers rather limited support to exact inference.

In contrast, WSPNs lend themselves naturally to efficient inference and learning as well as end-to-end training together with deep neural networks. They also have clear probabilistic semantics. In fact, WSPNs can be seen as generalized directed acyclic graphs (DAGs) of mixture models in the spectral domain, with sum nodes corresponding to mixtures over subsets of variables and product nodes corresponding to features or mixture components. Specifically, they consist of the following two ingredients: Whittle Likelihood and complex-valued SPNs.

Ingredient 1: Whittle Likelihood. The Whittle likelihood models the multivariate time series in the spectral domain. Let $X = [x(1), \dots, x(T)]$, with $x(t) \in \mathbb{R}^p$, be a realization of a p-dimensional (p-D) time series with length T. For

[1] Source code is available at: https://github.com/ml-research/WhittleNetworks

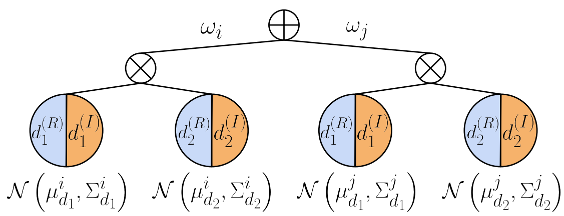

Figure 2. Illustration of a CoSPN modeling a density over two complex RVs d_1 and d_2. Leaf node encodes a 2-D Gaussian with mean $\mu_{d_k} \in \mathbb{R}^2$ and covariance matrix $\Sigma_{d_k} \in \mathbb{R}^{2\times 2}$ over the real and imaginary parts. The sum node $\bigoplus$ has a left (i) and a right branch (j), the corresponding weights are w_i and w_j. It computes the convex combination of its children.

$t \in \mathbb{Z}$, $x(t)$ is Gaussian stationary if:

$$E(x(t)) = \mu \quad \forall t \in \mathbb{Z} \tag{1}$$

$$\text{Cov}(x(t), x(t+h)) = \mathbf{\Gamma}(h) \quad \forall t, h \in \mathbb{Z}. \tag{2}$$

$X_{1:N} = \{X^1, \ldots, X^N\}$ are N independent realizations of the time series. In spectral domain of each sequence, $d_{n,k} \in \mathbb{C}^p$ denotes the discrete Fourier coefficient of the n^{th} sequence at frequency $\lambda_k = 2\pi k/T, k = 0, \ldots, T-1$:

$$d_{n,k} = T^{-1} \sum\nolimits_{t=0}^{T-1} x_n(t) e^{-i\lambda_k t}. \tag{3}$$

Based on the Whittle approximation assumption (Whittle, 1953), the Fourier coefficients are independent complex normal RVs with mean zero:

$$d_{n,k} \sim \mathcal{N}(0, S_k), \quad k = 0, \ldots, T-1, \tag{4}$$

where $S_k \in \mathbb{C}^{p\times p}$ is the *spectral density matrix*. For a stationary time series, its spectral density matrix is defined as:

$$S_k = \sum\nolimits_{h=-\infty}^{\infty} \mathbf{\Gamma}(h) e^{-i\lambda_k h}. \tag{5}$$

The Whittle likelihood of the N realizations is defined as: $p(X_{1:N} \mid S_{0:T-1}) \approx$

$$\prod\nolimits_{n=1}^{N} \prod\nolimits_{k=0}^{T-1} \frac{1}{\pi^p \, |S_k|} e^{-d^*_{n,k} S_k^{-1} d_{n,k}}. \tag{6}$$

The Whittle approximation holds asymptotically with large T and it has been used in the Bayesian context.

To overcome the limitations in Whittle likelihood inference with TGMs, Whittle Networks place an SPN (Poon & Domingos, 2011)—a tractable and expressive density estimator—across the frequencies.

Ingredient 2: Complex-Valued SPNs (CoSPNs). A complex random variable $d \in \mathbb{C}^p$ that follows complex normal distribution has its real and imaginary parts being jointly normal distributed. Based on the fact that the real and imaginary parts are coupled, it is a more sensible idea to model the real and imaginary parts as a pair of RVs in an SPN, which results in Complex-valued SPNs (CoSPNs). More formally, a CoSPN S over a set $\mathbf{D}$ of p complex-valued RVs is a probabilistic model defined via a DAG containing three types of nodes: *input distributions* (the *leaves*), *sums* and *products*. All leaves of the CoSPN are density functions over some subset $\mathbf{Q} \subseteq \mathbf{D}$ of pairs of real-valued RVs. Inner nodes are either weighted sums or products, denoted by S and P, respectively, i.e. $\mathsf{S} = \sum_{\mathsf{N}\in\mathbf{ch}(\mathsf{S})} \omega_{\mathsf{S},\mathsf{N}} \mathsf{N}$ and $\mathsf{P} = \prod_{\mathsf{N}\in\mathbf{ch}(\mathsf{P})} \mathsf{N}$, where $\mathbf{ch}(\cdot)$ denotes the children of a node. The sum weights $\omega_{\mathsf{S},\mathsf{N}}$ are assumed to be non-negative and normalized: $\omega_{\mathsf{S},\mathsf{N}} \geq 0, \sum_{\mathsf{N}} \omega_{\mathsf{S},\mathsf{N}} = 1$. CoSPNs make use of pairwise Gaussian leaf nodes for modeling the pair of real and imaginary parts of the complex RVs, assuming that the real and imaginary parts from one complex RV are correlated. The pairwise Gaussian leaf node is modeled using a vector of means $\mu_{d_k} \in \mathbb{R}^2$ and a covariance matrix $\Sigma_{d_k} \in \mathbb{R}^{2\times 2}$, as illustrated in Fig. 2. With the pairwise Gaussian leaf density of the complex RVs, CoSPN essentially encodes the joint density $p([d_1^{(R)}, d_1^{(I)}], \ldots, [d_p^{(R)}, d_p^{(I)}])$. In analogy to SPNs, the scope of an input distribution N in a CoSPN is defined as the set of RVs $\mathbf{Y}$ for which N is a distribution function, i.e. $\mathbf{sc}(\mathsf{N}) := \mathbf{Y}$. The scope of sum or product node N is recursively defined as $\mathbf{sc}(\mathsf{N}) = \bigcup_{\mathsf{N}'\in\mathbf{ch}(\mathsf{N})} \mathbf{sc}(\mathsf{N}')$.

To represent a valid probability density, CoSPNs should satisfy two structural constraints (Poon & Domingos, 2011), namely completeness and decomposability. Specifically, a CoSPN is *complete* if for each sum S it holds that $\mathbf{sc}(\mathsf{N}') = \mathbf{sc}(\mathsf{N}'')$, for all $\mathsf{N}', \mathsf{N}'' \in \mathbf{ch}(\mathsf{S})$. A CoSPN is *decomposable* if it holds for each product P that $\mathbf{sc}(\mathsf{N}') \cap \mathbf{sc}(\mathsf{N}'') = \emptyset$, for all $\mathsf{N}' \neq \mathsf{N}'' \in \mathbf{ch}(\mathsf{P})$. In that way, all nodes in a CoSPN recursively define a valid complex-valued distribution over their respective scopes: the leaves are complex-valued distributions by definition, sum nodes are mixtures of their children, and products are factorized, complex-valued distributions, assuming context-specific independence among the scopes of their children.

CoSPNs feature tractable probabilistic inference. For example, CoSPNs allow one to compute arbitrary marginal densities: In particular, let $S(\mathbf{x})$ be a density over $\mathbf{X}$ represented by CoSPN S, and let $\bar{\mathbf{X}} = \{X_{i_1}, \ldots, X_{i_M}\}$ be a set of RVs to be marginalized. The marginal density over $\mathbf{Z} = \mathbf{X} \setminus \bar{\mathbf{X}}$ can be computed as $S(\mathbf{Z}) = \int_{x_{i_1}} \ldots \int_{x_{i_M}} S(x_{i_1}, \ldots, x_{i_M}, \mathbf{Z}) \mathrm{d}x_{i_1} \ldots x_{i_M}$. The integrals can be iteratively swapped with sums and distributed over products in the CoSPN (Peharz et al., 2015).

Similar to Gens & Domingos (2013), learning a CoSPN can be done by clustering over instances, treating them as $2p$-D vectors, to learn sum nodes, and by non-parametric independence test over both real and imaginary components to learn a product node. For the pairwise Gaussian leaf case, the non-parametric independence test needs to be adapted, such that two complex RVs are dependent if any of the four combinations of the real and imaginary components show a sort of dependency. Random-and-Tensorized SPNs (RAT-SPNs) (Peharz et al., 2020b), instead, generate a random tree structure and then optimize the weights in a "classical deep learning manner", which enables us to adapt CoSPNs together with other deep architectures.

WSPNs = Whittle Likelihood + CoSPNs. With the Whittle Likelihood and CoSPNs at hand, we are now ready to introduce WSPNs: CoSPNs are used to build the joint complex normal distribution of the Fourier coefficients of the time series. Note that the Fourier coefficients from DFT for real-valued sequences are Hermitian-symmetric. Therefore the negative frequency coefficients are redundant, and in total only $T_W = \lfloor T/2 \rfloor + 1$ Fourier coefficients need to be modeled.

We know from the Whittle assumption that the independence of Fourier coefficients of different frequencies holds for stationary time series. In general, we assume that without the independent Fourier coefficients assumption, the joint distribution still models the time series in the Fourier domain, even with more flexibility in modeling both stationary and non-stationary time series. Therefore, in contrast to TGM limitations, two more constraints can be relaxed when modeling general time series:

- The mean of each frequency in (4) need not be 0.
- The Fourier coefficients of different frequencies need not be independent as in (6).

With the above two relaxations, the Fourier coefficients of both stationary and non-stationary general time series can be modeled with WSPNs. The Whittle likelihood of the first T_W Fourier coefficients from a general time series can be modeled with one WSPN: $p(X_{1:N} \mid Co) \approx$

$$\prod_{n=1}^{N} p(d_{n,0}, \dots, d_{n,T_W-1}) \stackrel{\text{def}}{=} \prod_{n=1}^{N} p(d_n), \qquad (7)$$

where Co denotes the structure and parameters of the CoSPN trained from data, $d_{n,k} \in \mathbb{C}^p$ denotes the k^{th} Fourier coefficients from the n^{th} sample from data, and $d_n \in \mathbb{C}^{p \times T_W}$ denotes the first T_W Fourier coefficients of one multivariate time series.

3. Opening the Blackbox of Whittle SPN

While WSPNs are tractable, they are just computational graphs and, hence, blackboxes w.r.t. the (conditional) independence relations. For p-D multivariate time series, there is great interest in finding the structure that represents the conditional independence among the p components in the form of graphs. We now show how to extract it from WSPNs.

The vanilla search-based method for structure learning of graphical models usually consists of a structure learning algorithm, e.g. Hill climbing (Herskovits, 1991; Gámez et al., 2011), and a score, e.g. MDL (Rissanen, 1983). For DAGs, if the directions of edges are predefined, the graph can be learned efficiently via a 3-phase algorithm presented in Cheng et al. (1997). More generally, one can employ an order swapping algorithm (Teyssier & Koller, 2005) without providing the order of the nodes.

Moving to Whittle likelihood, it was indeed proposed for Bayesian structure learning in Tank et al. (2015). First, T spectral density matrices $S_{0:T-1}$ are computed from time series $X_{1:N}$. As the Whittle likelihood $p(X_{1:N} \mid G, S_{0:T-1})$ can be factorized given graph G, it can be maximized by searching over graph structures. Feature-inclusion stochastic search (FINCS) (Scott & Carvalho, 2008) is then applied to search for G that maximizes the above likelihood.

With WSPN, we can proceed as follows. Denote the Fourier coefficients from a subset s of the p components of the time series as $d_n^{\{s\}}$, and $p(d_n^{\{s\}})$ the marginal density. Let $G = (V, E)$ be a decomposable graph with vertex set $V = \{v_1, \dots, v_p\}$ and edge set E. For DAGs, given each node v_i and its parents $Pa_G(v_i)$, and the corresponding CoSPN Co learned from time series $X_{1:N}$, (7) can be factorized based on the graph structure by chain rule: $p(X_{1:N} \mid G, Co) \approx$

$$\prod_{n=1}^{N} \frac{\prod_{v_i \in V} p(d_n^{\{v_i \cup Pa_G(v_i)\}} \mid Co)}{\prod_{v_i \in V} p(d_n^{\{Pa_G(v_i)\}} \mid Co)}. \qquad (8)$$

The factorization for undirected graph is similar, over cliques and separators (Tank et al., 2015). Thus, given time series $X_{1:N}$, we first learn a WSPN that models time series in the spectral domain. Then, we start from an empty graph G and add edges iteratively. To add an edge, we create a list of all possible edge candidates to form a graph, and compute the list of Whittle likelihoods from (8), given the obtained WSPN and each possible new graph. The edge that mostly increases the Whittle likelihood is added to G. We stop when adding an edge decreases the Whittle likelihood or when there are no more edge candidates. Bayesian information criterion (BIC) (Schwarz, 1978) can be naturally applied to handle the complexity of the graph.

4. Whittle Networks: Putting WSPNs onto the Deep Learning Stack

Like other PCs, WSPNs can be vectorized and used within GPU-supporting implementations (Peharz et al., 2020b; Trapp et al., 2019; Peharz et al., 2020a). Note that the discrete (fast) Fourier transformation (DFT) can naturally be vectorized, too. Thus, WSPNs are differentiable and can be trained end-to-end together with NNs, resulting in Whittle Networks.

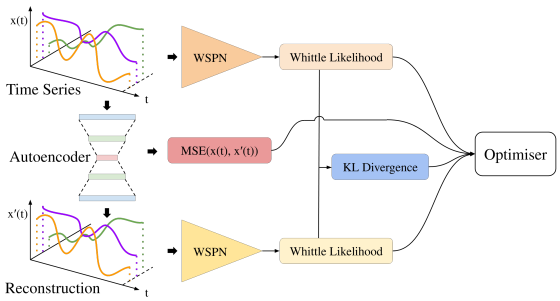

Figure 3. Whittle AE, an instantiation of Whittle Networks.

Therefore, we introduce Whittle AE, an instantiation of Whittle Networks, shown in Fig. 3. It combines a standard AE with a proper (deep) likelihood function for time series using WSPNs. To train the AE whose output distribution becomes similar to its input distribution, we employ two WSPNs alongside the AE. Each WSPN models the spectral domain of AE input and output, respectively. The distributions of the AE input and output in the spectral domain will be pushed closer by minimizing the Kullback-Leibler (KL) divergence between the distributions modeled by the two WSPNs. The combined loss consists of the reconstruction error, two negative Whittle likelihoods of both WSPNs, and the KL divergence between the WSPNs. In this way, we get a simple but effective way to equip AEs with densities and probabilistic inference. This can be used to assess how likely the AE reconstruction originates from the input distribution. Generally, we note that Whittle Networks can be combined with any other deep neural architecture, providing meaningful probabilities for time series.

5. Experimental Evaluation

Our intention here is to investigate the benefits of modeling time series with the proposed Whittle Networks. In particular, we investigated the following questions: **(Q1)** Do WSPNs capture densities over time series better than baselines? **(Q2)** Can WSPNs discover the conditional independence of time series? **(Q3)** Can Whittle Networks provide meaningful probabilities for deep neural networks?

Experimental Protocol and Datasets. Stationary time series is of great interest since the Whittle approximation which inspired us is based on the stationary assumption. Therefore, we use two real-world market datasets acquired from “Yahoo! Finance Data”. The first one is the index values of 11 sectors from “Standard & Poor’s” (*S&P*) from October 16, 2013 to May 24, 2019 (See Fig. 1 (Left)). The second one is the global stock index (*Stock*) from 17 markets extracted from June 2, 1997 to June 30, 1999. Both *S&P* and *Stock* datasets are applied first with log-return transformation, assuming them to be stationary (Stărică & Granger, 2005), and then a sliding window of size 32, ending up in 44 and 50 time series instances. Simulation data from Vector Autoregressive process (*VAR*) (Sims, 1980) is also used to discover the conditional independencies. Details of the three stationary datasets can be found in Appendix A.

With the aim of showing that Whittle Networks can deal with not only stationary time series but also with non-stationary processes, we employ the following synthetic and real-world datasets for non-stationary time series. The synthetic *Sine* data consists of 3 trigonometric sines with same frequency while different phases with Gaussian noise, 2 sine series with another frequency with different phases with Gaussian noise, and one series of pure Gaussian noise. We shuffle the order of time series components as out-of-domain (ood) samples. The synthetic *Billiards* data contains simulations of 6 trajectories from the horizontal and vertical locations of 3 balls, unnatural trajectories form the ood set. The synthetic *Mackey-Glass* series (Gers et al., 2002) consists of two channels and is used for forecasting test. Although MNIST (LeCun et al., 1998) is not a typical univariate time series dataset, it is widely used in prominent time series processing works (Van Oord et al., 2016; Esteban et al., 2017; Le et al., 2015). In this work, we use MNIST also to examine the ability of Whittle Networks in modeling time series with high dimensionality. Finally, we used hyperspectral images with 328 wavelengths of *plants* for a qualitative analysis of anomaly detection. Each vector from one pixel with length 328 can be viewed as a single univariate time series and we aim for detecting the unhealthy areas on the leaf. Fig. 5 (Top) shows RGB views of healthy and unhealthy leaves with dark spots as infected areas. Details of the non-stationary datasets are described in Appendix B.

(Q1) Modeling Time Series with WSPNs. We trained SPNs with Gaussian leaves employing LearnSPN (Gens & Domingos, 2013), ResSPNs (Ventola et al., 2020), and WSPNs on all datasets. For LearnSPN and ResSPN, no pairwise constraint is applied when estimating independencies, i.e., the real and imaginary parts from one complex RV are free to split. “-Pair” uses diagonal covariance matrices to model the real and imaginary parts independently, while “-2d” takes full covariance matrices to jointly model the pairs as bidimensional Gaussians. LearnSPN, WSPN-Pair, and WSPN-2d encompass and do structure learning, while the structure of ResSPN, ResWSPN-Pair, and ResWSPN-2d are randomly generated. Furthermore, in the spirit of having a neural likelihood gold standard, we make use of Masked Autoencoder for Distribution Estimation (MADE) (Germain et al., 2015) with Gaussian conditionals, as implemented in Papamakarios et al. (2017), to estimate the joint probability density of the Fourier coefficients. The number of hidden layers in MADE is set to 1 for all datasets, while the hidden units vary from 200 to 600, depending on the number of RVs in each dataset. Note that this comparison is

Table 1. Average training, test (the higher the better $\Uparrow$, and best values in bold) and ood data (the lower the better $\Downarrow$, and best values in bold) log-likelihoods. WSPNs have high likelihood on training and test data but low likelihood on ood data, with relatively a large gap.

		LearnSPN	WSPN-Pair	WSPN-2d	ResSPN	ResWSPN-Pair	ResWSPN-2d	MADE
Sine	train $\Uparrow$	-0.47	2.65	**6.67**	**-60.62**	-148.48	-135.94	-105.91
	test $\Uparrow$	-0.75	1.85	**5.75**	**-63.13**	-150.90	-138.86	-108.64
	ood $\Downarrow$	$-\infty$	$-\infty$	$-\infty$	**-5880.85**	-4010.04	-4227.18	-11646865.93
MNIST	train $\Uparrow$	256.11	272.84	**277.50**	249.47	**254.46**	**254.30**	336.03
	test $\Uparrow$	254.99	270.40	**274.42**	245.67	251.74	**252.54**	327.22
	ood $\Downarrow$	**125.19**	160.29	155.76	**204.93**	218.25	216.01	136.98
Billiards	train $\Uparrow$	54.73	63.75	**65.01**	-367.83	-318.10	**-213.13**	-204.23
	test $\Uparrow$	52.80	**54.14**	**54.12**	-377.38	-324.78	**-219.04**	-252.51
	ood $\Downarrow$	-1984.38	-2348.57	**-2435.70**	-1003.49	-1052.21	**-2113.68**	-89521.82
S&P	train $\Uparrow$	-191.64	113.06	**174.45**	308.22	194.57	**1831.91**	359.52
Stock	train $\Uparrow$	-615.76	328.90	**417.81**	257.03	496.07	**1172.85**	639.10

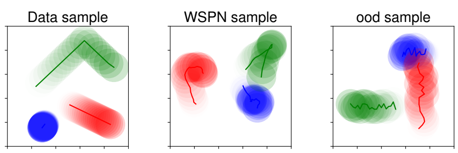

Figure 4. Billiards trajectories from the training set (Left), WSPN samples (Middle) and unnatural trajectories as ood samples (Right). Earlier time is demonstrated with more transparency, i.e., the ball starts from the most transparent side of the trajectory and moves towards the darker side.

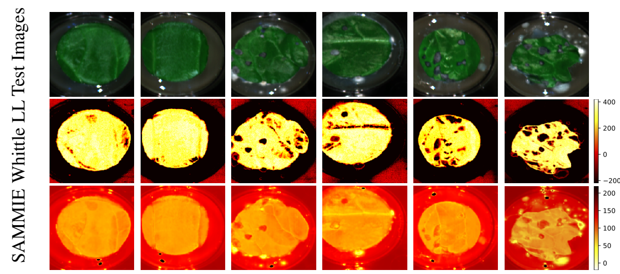

Figure 5. Anomaly detection within hyperspectral images. (Top) Visualization in RGB channels. (Middle) Heatmap showing the pixel-wise likelihood (log-scale) of WSPN. Pixels with higher value are more likely to belong to healthy areas. (Bottom) Heatmap of SAMMIE's reconstruction error (Kerner et al., 2019). Pixels with higher reconstruction error are considered as anomalies. The left two columns are healthy and the right four are inoculated.

not fair since MADEs are highly specialized for likelihood estimation and they cannot be easily adopted for computing general inference and efficient sampling due to the architectural constraints necessary for holding the autoregressive property.

As we can see from Tab. 1, WSPNs produce generally higher likelihood for training and test sets, compared to LearnSPN or ResSPN. Moreover, modeling the complex RV with a full covariance matrix in the leaf node ("-2d") provides higher likelihood for training/test set, and lower likelihood for ood set, compared with modeling them independently ("-Pair"). The likelihood of ood data from WSPN is also lower, except for MNIST data. The reason might be the image similarity given that all digits are centered. The differences of likelihoods are statistically significant according to a Wilcoxon signed-rank test with $p = 0.05$, "-Pair" and "-2d" perform equally better than ResSPN on MNIST and than LearnSPN on *Billiards*. Additionally, WSPNs achieve high likelihoods and it is competitive with our gold standard MADE. On MNIST, MADE provides higher likelihoods than WSPNs mainly because: 1) MADE is a strong and powerful density estimator. 2) MNIST is not a proper time series, since it has some "all zeros" rows as components in a multivariate series.

Fig. 4 shows trajectories of the 3 balls from *Billiards*. An example of real movement is illustrated in Fig. 4 (Left). Fig. 4 (Middle) shows trajectories after the inverse Fourier transform of the sampled Fourier coefficients from the WSPN trained on *Billiards*. The behaviours of straight-line-moving and rebound of the balls are successfully captured by the WSPN model. The movement of the green ball looks very realistic: it first moves towards the upper right corner, hits the wall, and then goes down to the left. The sample from *Billiards* shows the strong power of modeling the entire multivariate time series with WSPNs, without losing the characteristics of the original time series. To provide a comparison, ood trajectories are visualized in Fig. 4 (Right).

To illustrate this qualitatively on *plants*, Fig. 5 shows heatmaps of Whittle likelihood, as well as the reconstruction error maps got from SAMMIE (Kerner et al., 2019). As one can see, WSPN models the healthy pixels well and is able to give lower likelihood to pixels from infected or non-leaf areas, while SAMMIE fails to discover some infected spots.

Conditional SPNs (CSPNs) (Shao et al., 2020) can be applied to model the conditional distribution of $p(X_t \mid X_{t-1})$ in the time domain. By employing CSPNs as compo-

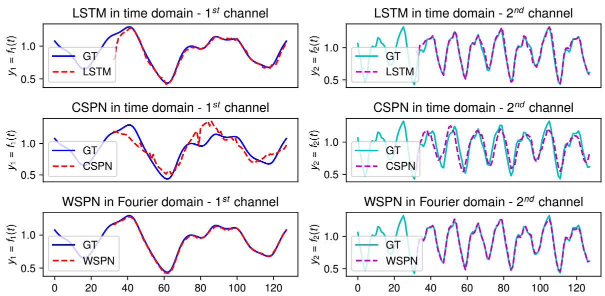

Figure 6. Predictions from LSTM, CSPN, WSPN, and ground truth (GT) on part of *Mackey-Glass* series.

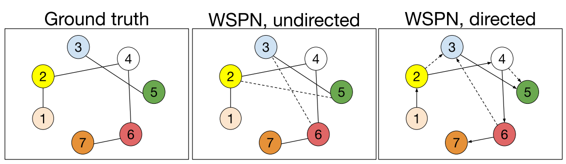

Figure 7. Extracting conditional independencies of *VAR* series. (Left) Ground truth. (Middle) Undirected graph from WSPN. (Right) Directed graph from WSPN. With BIC dashed edges are not generated and the final graphs perfectly match the ground truth.

nent, WSPNs can also model the conditional Whittle likelihood $p(d_t \mid d_{t-1})$ to perform forecasting straightforwardly. Given data from the past, one can compute predictions by means of most probable explanation (MPE) queries. We compared WSPN, CSPN, and LSTM as strong prediction baseline on *Mackey-Glass*. Details of the experimental settings are described in Appendix C. CSPN achieved a mean squared error (MSE) of 0.0119, WSPN 0.0022, and LSTM 0.0014 but with slower convergence. Qualitative results are shown in Fig. 6, one can see that WSPN performs better than CSPN and it has a competitive MSE with LSTM. Even if the core objective is to model the joint distribution of the whole time series rather than computing only predictions, WSPNs can also compute accurate forecasting. Thus, **(Q1)** can be answered affirmatively, WSPNs capture densities better than baselines.

(Q2) Conditional Independencies among Time Series. The conditional independencies of the *VAR* series are determined when it is created. We extract both directed and undirected graphs from WSPN trained on *VAR* series to visualize its conditional independencies. Fig. 7 shows that WSPN successfully extracted the true conditional independencies of a 7-D *VAR* series. Applying BIC helped to filter out the edges that increase the Whittle likelihood with a negligible increment.

Regarding real-world time series, the conditional independencies derived from WSPNs learned on *S&P* and *Stock* are shown in Fig. 1 and Fig. 8 respectively. The thickness of an

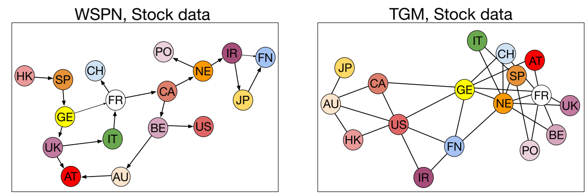

Figure 8. Extracting conditional independencies of *stock*. (Left) Directed independence structure among the 17 markets discovered by WSPN. (Right) Undirected independence structure derived with (Bayesian) TGM directly taken from Tank et al. (2015).

edge represents the increase of Whittle likelihood obtained by adding that particular edge. We set the maximum number of predecessors and successors to 2. For the comparison, we adopted hill climbing for learning (non-Bayesian) TGMs by using the OpenMarkov toolbox (Arias et al., 2019).

Regarding *S&P* data, the directed independence structure (DAG) shown in Fig. 1 (Middle) unveils interesting correlations among 11 sectors. WSPNs discover the correlations between, e.g., Information Technology and Financials, Industrials and Materials, Financials and Materials. Looking at the original time series in Fig. 1 (Left), one can observe that the indices of Materials and Financials seem to be quite synchronized, thus, the correlation between these two sectors makes sense. At the same time, the (non-Bayesian) TGM failed to discover some correlations, e.g., between Industrials, Energy, and Utilities, cf. Fig. 1 (Right).

WSPNs may also help to discover the conditional independencies (DAG) of the global stock market. From Fig. 8 (Left), it is clear that the central and western European countries (AT, UK, IT, FR, GE, CH, SP) are well clustered. Moreover, Germany, France, and the UK are among the core components of the European cluster. Most parts of the graph follow the geographic relationship, except for CA and BE. One explanation could be that WSPN discovers the hidden correlations of stock indices between those French-speaking countries. In contrast, (Bayesian) TGM (Right) correctly finds the correlations among Asia-Pacific markets (US, CA, HK, JP, AU), which are not discovered by WSPN. However, the graph structure of the European countries is less informative compared to the WSPN one. Both models fail to discover the correlation of JP and HK with others. This is indeed more difficult to discover because both are geographically far from the other markets.

Since the synthetic *Sine* dataset is relatively simple, both WSPN and non-Bayesian TGM produce the same DAG: two edges connecting the 3 components with the same frequency, another edge between the two components with another frequency, and the Gaussian noise being independent. The graph is relatively simple, thus, it is shown in Appendix D.

Overall, Tab. 2 summarizes the likelihoods of the condi-

Table 2. WSPNs provide higher log-likelihoods given the extracted graph. Thus, WSPNs discover better conditional independence structures (DAGs) than non-Bayesian TGM.

	S&P	Stock	Sine
WSPNs	**101.22**	**297.86**	**1.15**
non-Bayesian TGM	88.50	288.94	**1.15**

Table 3. Average log-likelihoods of MNIST data from Whittle Network. Low density values for instances that do not follow the training density are marked in bold.

	WSPN Input	WSPN Output
train	295.49	411.32
test	295.22	411.10
outlier1	**239.78**	401.54
outlier2	**48.84**	397.58

tional independence structures discovered via WSPNs. Both WSPNs and non-Bayesian TGMs achieve the same performance on *Sine*, as the discovered structures are actually the same. On *S&P* and *Stock*, however, WSPNs achieve better performance, i.e., the conditional independencies they discovered fit data better. From the results on various datasets, WSPN shows its ability to discover the conditional independencies from both stationary and non-stationary time series. In general, the results clearly provide an affirmative answer to **(Q2)**: WSPNs are able to discover the conditional independencies of time series, even better than baselines.

(Q3) Whittle Networks: Deep Likelihoods for (Deep) Neural Networks. Unfortunately, the probability distributions of deep neural networks may generally not be well-calibrated (Guo et al., 2017). In order to investigate the ability of WSPNs to provide meaningful and calibrated probabilities for deep neural networks, we considered the Whittle AE outlined before and shown in Fig. 3. It consists of two WSPNs and one AE in between. The AE consists of an MLP with the following number of neurons for each layer: $128 - 64 - 16 - 2 - 16 - 64 - 128$, using sigmoid as the activation function. We vectorized the WSPNs as described above using RAT-SPN (Peharz et al., 2020b). This way, they can easily be integrated and optimized end-to-end together with the AE. The RAT-SPN hyperparameters, see Peharz et al. (2020b) for details, can be found in Appendix E. We trained the Whittle AE on MNIST by taking each image row as one univariate time series, in other words, each image forms one multivariate series.

The average log-likelihood of our selected training and test data for the input WSPN and output WSPN, see Fig. 3, are summarized in Tab. 3. As one can clearly see, the input WSPN provides high likelihood for training and test data

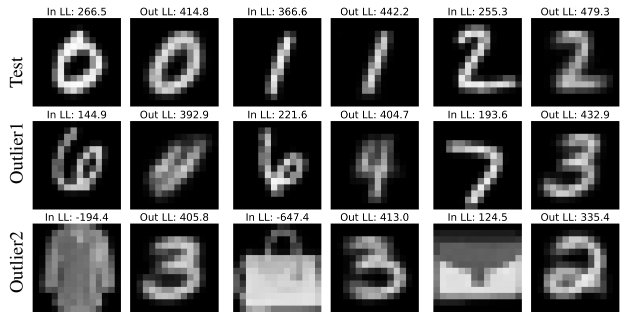

Figure 9. Qualitative results of Whittle AE. Visualization and likelihood (log-scale) of input (In) and output (Out) from (Top) MNIST *test* set (digits “0-4”), (Middle) **outlier1** (MNIST *test* set digits “5-9”), and (Bottom) **outlier2** (Fashion-MNIST *test* set). The Whittle AE provides higher likelihood to digits that are from the test set and lower likelihood to both outlier sets.

(digits “0-4”), while providing low likelihood for outlier1 (digits “5-9”). It is even lower for outlier2 (images from Fashion-MNIST (Xiao et al., 2017)), a very different domain. Surprisingly, the WSPN that models the output of the AE also produces relatively lower likelihood for outputs (reconstructions) when using “outliers” as input.

Generally speaking, unlikely outputs of deep neural networks, such as ood samples, can be detected by looking at the rather low likelihood provided by the WSPNs. Fig. 9 depicts various inputs and their corresponding outputs from the Whittle AE. As one can see, it is difficult to tell from the reconstructed images if the input is normal or rather an ood sample. However, the log-likelihoods of input and reconstructed output images clearly indicate that both ood sets have lower likelihood. For instance, the autoencoder wrongly reconstructs a pullover (first image of outlier2) as a digit “3” (second image of outlier2). Thus, both input and output WSPNs are able to recognize the ood and its reconstruction by assigning a low likelihood—especially on the input image, even when the reconstructed image looks like an authentic digit “3”. Although it is almost impossible to judge from the output image “3”, both likelihoods from input respectively output images illustrate that the input may be an ood sample and that the corresponding output is not trustworthy. More results from Whittle AE can be found in Appendix F. Overall, these results clearly provide an affirmative answer to **(Q3)**: Whittle Networks can provide meaningful probabilities for deep neural networks.

6. Conclusion

We introduced the first complex-valued SPN, called CoSPN, tailored for complex-valued normal distribution. In particular, using CoSPNs and the Whittle likelihood approximation, we proposed the first probabilistic circuits for multivariate time series, modeling temporal information as well as the conditional independence of multivariate time series implicitly in the spectral domain. Being able to compute the likelihood of time series in a tractable fashion is critical in developing practical and scalable likelihood functions for deep neural networks of time series. Our experimental results demonstrated that the resulting Whittle Networks can indeed provide meaningful probabilistic measures for deep neural networks. Providing a natural measure of uncertainty makes deep neural networks easier to interpret and to use for non-AI-experts, in particular when it comes to decision-making. Exploring this for other deep architectures is the most interesting avenue for future work, but one should also investigate more advanced structure learning methods for complex-valued SPNs in general. Regarding complex values, one could bring the idea of CoSPNs to other models like normalizing flows or wavelet networks. Furthermore, based on the introduced conditional WSPN for forecasting, we envision the exploration of dynamic WSPN or Whittle RNN as interesting future directions.

Acknowledgements

The authors thank the anonymous reviewers for their valuable feedback. This work was supported by the Federal Ministry of Education and Research (BMBF; project "MADESI", FKZ 01IS18043B, and Competence Center for AI and Labour; "kompAKI", FKZ 02L19C150), the German Science Foundation (DFG, German Research Foundation; GRK 1994/1 "AIPHES"), the Hessian Ministry of Higher Education, Research, Science and the Arts (HMWK; projects "The Third Wave of AI" and "The Adaptive Mind"), and the Hessian research priority programme LOEWE within the project "WhiteBox".

References

Arias, M., Prez-Martn, J., Luque, M., and Dez, F. J. Openmarkov, an open-source tool for probabilistic graphical models. In *Proc. of IJCAI*, pp. 6485–6487, 2019.

Bach, F. R. and Jordan, M. I. Learning graphical models for stationary time series. *IEEE Transactions on Signal Processing*, 52(8):2189–2199, 2004.

Buesing, L., Weber, T., Racaniere, S., Eslami, S., Rezende, D., Reichert, D. P., Viola, F., Besse, F., Gregor, K., Hassabis, D., and Wierstra, D. Learning and querying fast generative models for reinforcement learning. *arXiv preprint arXiv:1802.03006*, 2018.

Cheng, J., Bell, D., and Liu, W. Learning bayesian networks from data: an efficient approach based on information theory. In *Proc. of ACM CIKM*, 1997.

Connor, J. T., Martin, R. D., and Atlas, L. E. Recurrent neural networks and robust time series prediction. *IEEE Transactions on Neural Networks*, 5(2):240–254, 1994.

Dahlhaus, R. Graphical interaction models for multivariate time series. *Metrika*, 51(2):157–172, 2000.

Esteban, C., Hyland, S. L., and Rätsch, G. Real-valued (medical) time series generation with recurrent conditional gans. *arXiv preprint arXiv:1706.02633*, 2017.

Gámez, J. A., Mateo, J. L., and Puerta, J. M. Learning bayesian networks by hill climbing: efficient methods based on progressive restriction of the neighborhood. *Data Mining and Knowledge Discovery*, 22(1-2):106–148, 2011.

Gens, R. and Domingos, P. Learning the Structure of Sum-Product Networks. In *Proc. of ICML*, pp. 873–880, 2013.

Germain, M., Gregor, K., Murray, I., and Larochelle, H. Made: Masked autoencoder for distribution estimation. In *Proc. of ICML*, pp. 881–889, 2015.

Gers, F. A., Schmidhuber, J., and Cummins, F. Learning to forget: continual prediction with lstm. *Neural Computation*, 12(10):2451, 2000.

Gers, F. A., Eck, D., and Schmidhuber, J. Applying lstm to time series predictable through time-window approaches. In *Neural Nets WIRN Vietri-01*, pp. 193–200. Springer, 2002.

Guo, C., Pleiss, G., Sun, Y., and Weinberger, K. Q. On calibration of modern neural networks. In *Proc. of ICML*, pp. 1321–1330, 2017.

Herskovits, E. *Computer-based probabilistic-network construction*. PhD thesis, Stanford University USA, 1991.

Kalchbrenner, N., van den Oord, A., Simonyan, K., Danihelka, I., Vinyals, O., Graves, A., and Kavukcuoglu, K. Video pixel networks. In *Proc. of ICML*, pp. 1771–1779, 2017.

Kerner, H. R., Wellington, D. F., Wagstaff, K. L., Bell, J. F., Kwan, C., and Amor, H. B. Novelty detection for multispectral images with application to planetary exploration. In *Proc. of AAAI*, volume 33, pp. 9484–9491, 2019.

Koskela, T., Lehtokangas, M., Saarinen, J., and Kaski, K. Time series prediction with multilayer perceptron, fir and elman neural networks. In *Proc. of the World Congress on Neural Networks*, pp. 491–496, 1996.

Kossen, J., Stelzner, K., Hussing, M., Voelcker, C., and Kersting, K. Structured object-aware physics prediction for video modeling and planning. In *Proc. of ICLR*, 2020.

Krishnan, R. G., Shalit, U., and Sontag, D. A. Structured inference networks for nonlinear state space models. In *Proc. of AAAI*, pp. 2101–2109, 2017.

Le, Q. V., Jaitly, N., and Hinton, G. E. A simple way to initialize recurrent networks of rectified linear units. *arXiv preprint arXiv:1504.00941*, 2015.

LeCun, Y. and Bengio, Y. Convolutional networks for images, speech, and time series. *The handbook of brain theory and neural networks*, 3361(10), 1995.

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

Melibari, M., Poupart, P., Doshi, P., and Trimponias, G. Dynamic sum product networks for tractable inference on sequence data. In *Proc. of International Conference on PGM*, pp. 345–355, 2016.

Papamakarios, G., Pavlakou, T., and Murray, I. Masked autoregressive flow for density estimation. In *Proc. of NIPS*, pp. 2335–2344, 2017.

Peharz, R., Tschiatschek, S., Pernkopf, F., and Domingos, P. On theoretical properties of sum-product networks. *The Journal of Machine Learning Research*, 2015.

Peharz, R., Lang, S., Vergari, A., Stelzner, K., Molina, A., Trapp, M., Broeck, G. V. d., Kersting, K., and Ghahramani, Z. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *Proc. of ICML*, 2020a.

Peharz, R., Vergari, A., Stelzner, K., Molina, A., Shao, X., Trapp, M., Kersting, K., and Ghahramani, Z. Random sum-product networks: A simple and effective approach to probabilistic deep learning. In *Proc. of UAI*, pp. 334–344, 2020b.

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

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. In *Proc. of NeurIPS*, pp. 7785–7794, 2018.

Rissanen, J. A universal prior for integers and estimation by minimum description length. *The Annals of Statistics*, pp. 416–431, 1983.

Schwarz, G. Estimating the dimension of a model. *The Annals of statistics*, 6(2):461–464, 1978.

Schweitzer, F., Fagiolo, G., Sornette, D., Vega-Redondo, F., Vespignani, A., and White, D. R. Economic networks: The new challenges. *Science*, 325(5939):422–425, 2009.

Scott, J. G. and Carvalho, C. M. Feature-inclusion stochastic search for gaussian graphical models. *Journal of Computational and Graphical Statistics*, 17(4):790–808, 2008.

Shao, X., Molina, A., Vergari, A., Stelzner, K., Perharz, R., Liebig, T., and Kersting, K. Conditional sum-product networks: Imposing structure on deep probabilistic architectures. In *Proc. of International Conference on PGM*, 2020.

Sims, C. A. Macroeconomics and reality. *Econometrica: journal of the Econometric Society*, pp. 1–48, 1980.

Stărică, C. and Granger, C. Nonstationarities in stock returns. *Review of economics and statistics*, 87(3):503–522, 2005.

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

Tank, A., Foti, N. J., and Fox, E. B. Bayesian structure learning for stationary time series. In *Proc. of UAI*, pp. 872–881, 2015.

Teyssier, M. and Koller, D. Ordering-based search: A simple and effective algorithm for learning bayesian networks. In *Proc. of UAI*, pp. 584–590, 2005.

Trapp, M., Peharz, R., Ge, H., Pernkopf, F., and Ghahramani, Z. Bayesian learning of sum-product networks. In *Proc. of NeurIPS*, pp. 6344–6355, 2019.

Trapp, M., Peharz, R., Pernkopf, F., and Rasmussen, C. E. Deep structured mixtures of gaussian processes. In *Proc. of AISTATS*, pp. 2251–2261, 2020.

Van Oord, A., Kalchbrenner, N., and Kavukcuoglu, K. Pixel recurrent neural networks. In *Proc. of ICML*, pp. 1747–1756, 2016.

Ventola, F., Stelzner, K., Molina, A., and Kersting, K. Residual sum-product networks. In *Proc. of International Conference on PGM*, 2020.

Whittle, P. The analysis of multiple stationary time series. *Journal of the Royal Statistical Society: Series B (Methodological)*, 15(1):125–139, 1953.

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