Predictive Whittle Networks for Time Series

Abstract

Recent developments have shown that modeling in the spectral domain improves the accuracy in time series forecasting. However, state-of-the-art neural spectral forecasters do not generally yield trustworthy predictions. In particular, they lack the means to gauge predictive likelihoods and provide uncertainty estimates. We propose predictive Whittle networks to bridge this gap, which exploit both the advances of neural forecasting in the spectral domain and leverage tractable likelihoods of probabilistic circuits. For this purpose, we propose a novel Whittle forecasting loss that makes use of these predictive likelihoods to guide the training of the neural forecasting component. We demonstrate how predictive Whittle networks improve real-world forecasting accuracy, while also allowing a transformation back into the time domain, in order to provide the necessary feedback of when the model's prediction may become erratic.

1 INTRODUCTION

Time series modeling and forecasting have been a crucial area of research in machine learning, forming a prominent role in its application to several high-impact real-world problems, such as ecological modeling [Recknagel, 2001], finance [Dingli and Fournier, 2017] and healthcare [Alaa and van der Schaar, 2019]. Recent extensions of recurrent neural networks (RNN) [Rumelhart et al., 1985] can achieve impressive performance on complex multivariate time series. However, in many real-world applications, time series are highly subject to several influence factors, which are often hard to capture [Stankevičiūtė et al., 2021]. For example, influence factors could be, or strongly depend on, complex phenomena such as weather conditions or extreme events like natural calamities or a pandemic. In these cases, the model's forecasts will likely be less accurate. To properly detect such scenarios, a confidence score of the prediction is valuable [Guo et al., 2017] which can make the predictions trustworthy and better support the users in decision-making processes. Whereas several approaches exist, mostly based on Gaussian processes that can also quantify the predictive uncertainty [Seeger, 2004, Rasmussen and Williams, 2006], they are computationally expensive [Bruinsma et al., 2020]. Although one can use hybrid models to scale [Trapp et al., 2020] or add stricter constraints [Corani et al., 2021], these solutions are usually less accurate than current neural counterparts [Alpay et al., 2016]. Recent neural models that operate in the time domain have tackled time series forecasting from a probabilistic perspective, e.g. by making use of neural density estimators [Rasul et al., 2021a] or by employing auto-regressive denoising diffusion models [Rasul et al., 2021b]. These models can be either slow in sample generation or in likelihood computation due to their mostly auto-regressive nature. Moreover, they do not provide a confidence score or a likelihood for a sequence composed of a prediction and its context. Such a measure would enable users to quickly detect potentially problematic forecasts.

Recently, it has been shown that modeling time series in the spectral domain is beneficial in both forecasting accuracy and efficiency since the spectral representation of a time series is generally more compact [Wolter et al., 2020]. Despite being accurate and efficient, current neural spectral time series forecasters do not provide any likelihood score or uncertainty estimate of their predictions in the time domain, nor for an entire sequence like a context with a prediction. On the other hand, although previous probabilistic spectral methods like Tank et al. [2015] and Yu et al. [2021a] have shown improved performance in capturing the distribution of a multivariate time series, they do not tackle forecasting and their predictive power does not outperform established neural architectures. In general, the missing ability to gauge predictive likelihoods is important since they can be exploited during training to learn more accurate forecasters.

Motivated by these prior works, we introduce predictive

Submitted to the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022). **To be used for reviewing only**.

Whittle networks, which integrate a neural spectral forecaster, such as Spectral RNN [Wolter et al., 2020] or a spectral Transformer variant, with Whittle probabilistic circuits (Whittle PCs) [Yu et al., 2021a], i.e. tractable probabilistic models which make use of the Whittle approximation [Whittle, 1953] to facilitate the modeling of the Fourier coefficients of a time series. The aim of predictive Whittle network is to integrate the powerful predictive accuracy of neural spectral forecasters with the useful feedback from tractable and flexible density estimators, in our case, Whittle PCs. We make the following key contributions:

- We propose predictive Whittle networks and the Whittle forecasting loss to exploit the predictive power of spectral neural forecasters and gauge tractable likelihoods from a probabilistic circuit to improve forecasting accuracy.
- We introduce a novel log-likelihood ratio score to provide predictive uncertainty estimates in the time domain based on likelihoods from the spectral domain.

Moreover, to better suit in predictive Whittle networks, we devise improved variants for the neural and probabilistic components.

2 RELATED WORK

In a simplified picture, approaches that predict the future course of a time series can be categorized as relying on black-box neural network models, or constructing an elaborate probabilistic model to capture the statistical dependencies among the series' random variables. Intuitively, these perspectives seem to trade-off prediction performance with the ability to accurately gauge data likelihood. We aim to leverage the benefits of both of these views in our work.

Probabilistic Modeling of Time Series: A well-known approach to forecasting is to leverage a probabilistic machine learning perspective. For instance, the popular Gaussian processes (GPs) compute probabilistic non-linear regression, allowing exact posterior inference and a natural computation of predictive uncertainty. GPs have been intensively explored for time series regression, classification [Rasmussen and Williams, 2006, Nickisch and Rasmussen, 2008], and have recently been revisited for time series forecasting [Sun et al., 2014, Corani et al., 2021]. Given that GPs do not scale easily, it has been proposed to scale them by employing probabilistic hierarchical mixtures, both for uni-variate [Trapp et al., 2020] and multi-output regression [Yu et al., 2021b].

Alternative generative models which use well-defined likelihood loss functions have thus been proposed. On the one hand, Rangapuram et al. [2018] combined state space models with deep neural networks, while it only centered on forecasting, without modeling the joint distribution of the entire time series. On the other hand, sum-product networks (SPNs) [Poon and Domingos, 2011], a member of the probabilistic circuit (PC) family, have previously been investigated for time series modeling, e.g. dynamics SPNs [Melibari et al., 2016] and the later extension recurrent SPNs [Kalra et al., 2018]. Whereas these approaches now provide tractable and exact probabilistic inference, they have limited representational power because of their strict structural constraints. Thus, they are not as accurate forecasters as deep neural models.

Neural Spectral Forecasting: Recurrent neural architectures, such as long short-term memory [Hochreiter and Schmidhuber, 1997] and gated recurrent unit (GRU) [Cho et al., 2014] networks, have paved the way for more accurate neural forecasting. In several scenarios, these approaches have been shown to outperform traditional non-neural models [Siami-Namini et al., 2018]. For instance, N-BEATS [Oreshkin et al., 2019] has achieved great performance on various challenging data sets. Transformers [Vaswani et al., 2017] have been investigated to further improve this forecasting ability in the time domain [Li et al., 2019], with Informer [Zhou et al., 2021] setting the new state-of-the-art at the price of an enormous increase of model size. In a similar spirit, neural auto-regressive models and normalizing flows have been shown to improve predictions [Rasul et al., 2021a, Salinas et al., 2020, Rasul et al., 2021b], but could be difficult to train or slow due to their auto-regressive nature. Recently, spectral RNN [Wolter et al., 2020] has demonstrated that it is beneficial to transform the time series into the spectral domain, in order to obtain a compact and efficient representation that fosters modeling capabilities and yields further performance enhancements. Such spectral modeling has also been pursued in neural sequence prediction with the complex Transformer [Yang et al., 2020]. However, these neural spectral methods do not provide the likelihood of the data, nor of their predictions.

Probabilistic Spectral Forecasting: Tank et al. [2015] have introduced a probabilistic approach that works on a spectral representation of stationary time series. They make use of the Whittle approximation to estimate the structure of a graphical model, which encodes the dependencies between the time series components. The Whittle approximation has further been employed in Whittle networks [Yu et al., 2021a], which aim to model the joint distribution of more general non-stationary time series. Whittle networks pose Whittle PCs on top of neural models to inspect their behavior and to capture complex dependencies among the time series components in the spectral domain. They provide the likelihood of an entire time series in the spectral domain but it can not be transformed directly to point-wise likelihoods in the time domain.

In our work, we build on top of the recent advances in all three above lines of research. We propose a hybrid approach that leverages recent insights from modeling in the spectral domain and combines the benefits of neural forecasters with

those of PCs. In this way, we are able to obtain tractable likelihoods and gauge them to further guide training to improve the predictive accuracy.

3 PREDICTIVE WHITTLE NETWORKS

In this section, we introduce the predictive Whittle network (PWN). It takes advantage of two distinct elements, namely, a neural spectral forecaster and a Whittle PC, to improve forecasting accuracy and provide useful uncertainty estimates for its predictions. PWN leverages the predictive power of the neural element and gauges the likelihoods from the probabilistic element to weigh its predictions. This is achieved with the Whittle forecasting loss, described in Section 3.1. Then, for each element, we introduce two variants better suited for spectral modeling. Thanks to the flexibility of PWN, they can be used interchangeably. The variants for the neural element are discussed in Section 3.2, while the ones for the probabilistic element are discussed in Section 3.3. A graphical representation of the architecture is shown in Fig. 1. Moreover, in Section 3.4, we present a novel score to provide predictive uncertainties in the time domain. Having such estimates in the time domain is essential to provide intelligible feedback on the predictions.

3.1 WHITTLE FORECASTING LOSS & TRAINING

We introduce the Whittle forecasting loss (WFLoss) to gauge likelihood estimates to guide the training of the neural component to superior predictive performance. As represented in Fig. 1, the loss is the connecting element between the neural and the probabilistic components of PWN.

Thanks to its inference capabilities, the Whittle PC can compute the conditional likelihood $\ell(\mathbf{y} \mid \mathbf{x})$ where $\mathbf{y}$ is a prediction and $\mathbf{x}$ its context. Therefore, to gauge the likelihoods provided for the predictions of the neural forecaster, we propose the WFLoss

$$\mathrm{WFLoss}(\mathbf{x}, \mathbf{y}_{Pred}, \mathbf{y}_{GT}) = \frac{1}{M}\sum_{i=0}^{M}(\mathbf{y}^i_{GT} - \mathbf{y}^i_{Pred})^2 \cdot (\ell^{\max} - \ell_{norm}(\mathbf{y}^i_{Pred} \mid \mathbf{x}^i)), \tag{1}$$

where

$$\ell_{norm}(\mathbf{y}^i_{Pred} \mid \mathbf{x}^i) = \frac{\ell(\mathbf{y}^i_{Pred} \mid \mathbf{x}^i) - \ell^{\max}}{\frac{1}{M}\sum_{j=1}^{M}(\ell(\mathbf{y}^j_{Pred} \mid \mathbf{x}^j) - \ell^{\max})}, \tag{2}$$

$\ell^{\max} = \max_k \ell(\mathbf{y}^k_{Pred} \mid \mathbf{x}^k)$, $\mathbf{y}_{Pred}$ denotes the model's prediction while $\mathbf{y}_{GT}$ denotes the ground truth, and

$$\mathrm{MSE}(\mathbf{y}_{GT}, \mathbf{y}_{Pred}) = \frac{1}{M}\sum_{i=0}^{M}(\mathbf{y}^i_{GT} - \mathbf{y}^i_{Pred})^2 \tag{3}$$

is the mean squared error (MSE). Following Eq. (1), the mean of $\ell_{norm}(\mathbf{y}^i_{Pred} \mid \mathbf{x}^i)$ in a mini-batch equals to 1, hence, the magnitude of the MSE-loss will not be influenced. WFLoss weighs the MSE based on the likelihood computed by the Whittle PC, so that samples with a low likelihood, that are rather in the tails of the distribution, are weighted less than those with a high likelihood. Therefore, our loss formulation prevents the neural spectral forecaster to fit outliers in the data and shifts the focus to data samples that follow the general distribution.

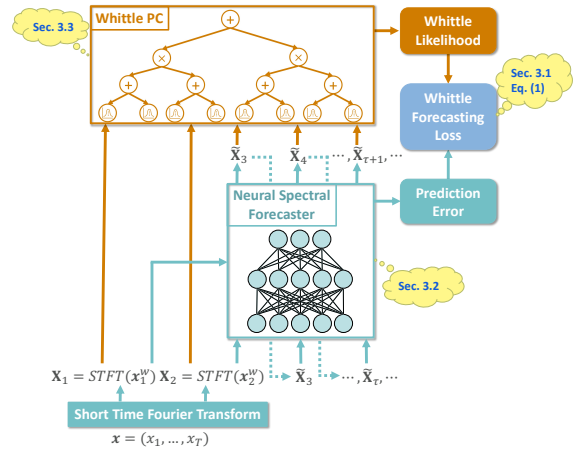

Figure 1: Overview of the predictive Whittle network architecture. The context $\mathbf{x}$ is transformed by STFT with a window w, a) flowing as context to the Whittle PC, b) serving as input to the neural spectral forecaster, resulting in the prediction of the Fourier coefficients $\widetilde{\mathbf{X}}_\tau$. These are then provided to the Whittle PC, which uses them with the context to compute the Whittle likelihood. Gauging these likelihood values during training improves forecasting accuracy.

The predictive Whittle network is then trained end-to-end in a coordinate descent fashion. In each optimization step, the weights of the Whittle PC are updated first by maximizing the likelihood, while the neural spectral forecaster's weights remain fixed. Afterwards, the Whittle PC weights are fixed and the neural spectral forecaster is optimized by minimizing the WFLoss. Details and a graphical representation of this alternating procedure can be found in Appendix A.

While training, the Whittle PC may require some epochs until its feedback is valuable for the neural spectral forecaster. Therefore, we employ a warm-up phase for it, by increasing $\beta \in [0, 1]$ linearly from 0 in the combined loss

$$\mathrm{Loss}(\beta, \mathbf{x}, \mathbf{y}_{Pred}, \mathbf{y}_{GT}) = (1-\beta)\,\mathrm{MSE}(\mathbf{y}_{GT}, \mathbf{y}_{Pred}) + \beta\,\mathrm{WFLoss}(\mathbf{x}, \mathbf{y}_{Pred}, \mathbf{y}_{GT}). \tag{4}$$

3.2 THE NEURAL ELEMENT: SPECTRAL FORECASTER

Here, we present two variants of the neural element which we tailor for spectral forecasting. In Fig. 1, this element is

represented as "Neural Spectral Forecaster".

Spectral RNN (SRNN) performs recurrent steps over windows retrieved from the short time Fourier transform (STFT) [Wolter et al., 2020]. Details of STFT ($\mathcal{F}_\mathcal{S}$) and its inverse iSTFT (${\mathcal{F}_\mathcal{S}}^{-1}$) can be found in Appendix B. Therefore, for a window $\mathbf{x}^w$ with width T_w and a step size S, it only has to perform $n_s = (T - T_w)/S + 3$ instead of the typical T time steps for a time series $\mathbf{x} = [x_1, x_2, \cdots x_T]$ of length T. The SRNN is defined as follows:

$$\begin{aligned} \mathbf{X}_\tau &= \mathcal{F}_\mathcal{S}(\mathbf{x}_\tau^w) \\ \mathbf{z}_\tau &= \mathbf{W}_c \mathbf{h}_{\tau-1} + \mathbf{V}_c \mathbf{X}_\tau + \mathbf{b}_c \\ \mathbf{h}_\tau &= f_a\left(\mathbf{z}_\tau\right) \\ \mathbf{y}_\tau &= {\mathcal{F}_\mathcal{S}}^{-1}(\mathbf{W}_{pc}\mathbf{h}_0, \ldots, \mathbf{W}_{pc}\mathbf{h}_\tau), \end{aligned} \tag{5}$$

with $\tau = [0, n_s]$ enumerating the total number of windows n_s. $\mathbf{W}_c$, $\mathbf{V}_c$, $\mathbf{b}_c$ and $\mathbf{W}_{pc}$ are weight matrices and $\mathbf{h}_\tau$ is the hidden state. Denote n_f the number of frequencies passing the low-pass filter in STFT. $\mathbf{X}_\tau \in \mathbb{C}^{n_f \times 1}$ is complex-valued, therefore, the RNN cell either needs to operate in the complex space or needs to provide projections $\mathcal{I} : \mathbb{C}^{n_f} \mapsto \mathbb{R}^{n_i}$, $\mathcal{O} : \mathbb{R}^{n_o} \mapsto \mathbb{C}^{n_f}$ for n_i-dimensional in- and n_o-dimensional outputs respectively.

According to our preliminary experiments (illustrated in Appendix C) and to what has been analyzed in Wolter et al. [2020], operating in the complex space is not substantially beneficial in terms of accuracy for the SRNN. Thus, we employ standard GRU [Chung et al., 2014] with projections. For projections, we use concatenation and splitting respectively, i.e. $\mathcal{I}(\mathbf{X}_\tau) = (Re(\mathbf{X}_\tau), Im(\mathbf{X}_\tau))$ and $\mathcal{O}(\mathbf{h}_\tau) = \mathbf{h}_\tau^{1,\ldots,n_f} + \mathbf{h}_\tau^{n_f+1,\ldots,2\times n_f} \cdot i$, where $n_i = n_o = 2 \times n_f$. Thus, $\mathbf{h}_\tau \in \mathbb{R}^{n_h \times 1}$, $\mathbf{W}_c \in \mathbb{R}^{n_h \times n_h}$, $\mathbf{V}_c \in \mathbb{R}^{n_h \times 2n_f}$, $\mathbf{b}_c \in \mathbb{R}^{n_h \times 1}$ and $\mathbf{W}_{pc} \in \mathbb{R}^{2n_f \times n_h}$, where n_h is the size of the hidden state. During our preliminary experiments, we have further discovered architectural improvements, i.e. we add residual links [He et al., 2016] to make the network deeper with 2 layers and apply dropout with $p = 0.1$. Details can be found in Appendix C.

Spectral Transformer (STransformer) is an architecture tailored for predicting time series in the spectral domain. It is based on the complex Transformer [Yang et al., 2020] which is designed for modeling complex-valued sequences (e.g. Fourier coefficients). However, some of the operations between complex values are only "emulated", e.g. the multi-head attention is emulated with 8 different real-valued attentions between real and imaginary parts.

We propose STransformer as an approach that works natively and holistically on complex numbers. Inputs of the model are the Fourier coefficients given by STFT. We apply positional encoding (PE) per window to preserve the correlation of adjacent frequencies:

$$\mathbf{X}_\tau = \mathcal{F}_\mathcal{S}(\mathbf{x}_w^\tau) + \mathrm{PE}^\tau, \tag{6}$$

where PE is defined as in [Vaswani et al., 2017]:

$$\mathrm{PE}_j^\tau = \begin{cases} \sin(\tau/1000^{2j/d_{\text{model}}}), & \text{if } j \mod 2 = 0 \\ \cos(\tau/1000^{2j/d_{\text{model}}}), & \text{else,} \end{cases} \tag{7}$$

where d_{model} denotes the embedding dimension of the Transformer, and j is the dimension of the positional encoding. To compute $\mathrm{Attention}(Q, K, V) = \mathrm{softmax}^c\left(\frac{QK^T}{\sqrt{d_k}}\right)V$, we shift all computations to the complex space, while employing an alternative softmax in a split-complex fashion [Wolter and Yao, 2018]:

$$\mathrm{softmax}^c(X) = \mathrm{softmax}(\mathrm{Re}(X)) + \mathrm{softmax}(\mathrm{Im}(X))i, \tag{8}$$

which allows the attention to be distributed over the real and imaginary parts separately. Analogously, we employ complex ReLU [Trabelsi et al., 2018]:

$$\mathrm{cReLU} = \max(0, \mathrm{Re}(X)) + \max(0, \mathrm{Im}(X))i. \tag{9}$$

For the output $\mathbf{y}_\tau$, we alter the decoding process to allow proper forecasting:

$$\mathbf{h}_\tau = \mathbf{dec}((\mathbf{X}_\tau, \mathbf{h_0}, ..., \mathbf{h_{\tau-1}})^T, \mathbf{enc}(\mathbf{X}_{0:\tau-1})), \tag{10}$$

$$\mathbf{y}_\tau = {\mathcal{F}_\mathcal{S}}^{-1}(\mathbf{W_d}\mathbf{h_0}, ..., \mathbf{W_d}\mathbf{h}_\tau). \tag{11}$$

The output $\mathbf{y}_\tau$ is computed based on all present and past decoding outputs $\mathbf{h}_\tau$, while **enc** and **dec** denote the encoding and decoding stacks respectively. Thus, now all operations are performed in the complex space. More details on our STransformer and respective preliminary experiments are provided in Appendix D.

3.3 THE PROBABILISTIC ELEMENT: WHITTLE PROBABILISTIC CIRCUIT

The Whittle approximation [Whittle, 1953] indicates that the Fourier coefficients of each frequency from a stationary time series are independently complex normal distributed. Recently, Yu et al. [2021a] extended the Whittle approximation to non-stationary time series by introducing the tractable density estimator called Whittle PCs. We use Whittle PCs as the probabilistic element of the PWN, as depicted in Fig. 1. Here, we consider two variants.

Conditional Whittle SPN (CWSPN) has been proposed in Yu et al. [2021a]. To provide a measure of how good a prediction ($\mathbf{y}$) is with respect to a context ($\mathbf{x}$), we aim to model the conditional Whittle likelihood $\ell(\mathbf{y} \mid \mathbf{x})$. Instead of the box window for discrete Fourier transform, in this work, we employ STFT for CWSPNs. Then, the input for the leaves of the CWSPN are the Fourier coefficients of $\mathbf{y}$ in the τth window at frequency k, i.e., $\mathbf{Y}_\tau^k = \mathcal{F}_\mathcal{S}(\mathbf{y})_\tau^k$. To account for the correlations between the real and imaginary parts, they are jointly modeled with a single pairwise Gaussian leaf node, parameterized by a vector of means $\mu_{\mathbf{Y}_\tau^k} \in \mathbb{R}^2$ and

a covariance matrix $\Sigma_{\mathbf{Y}_\tau^k} \in \mathbb{R}^{2\times 2}$. Thus, CWSPN encodes the conditional

$$p(d_1^1, \ldots, d_1^{n_f}, \ldots, d_{n_s}^1, \ldots, d_{n_s}^{n_f} \mid \mathcal{F}_\mathcal{S}(\mathbf{x})), \quad (12)$$

where $d_\tau^k = [\mathrm{Re}(\mathbf{Y}_\tau^k), \mathrm{Im}(\mathbf{Y}_\tau^k)]$. Then, based on Eq. (12), we define the conditional Whittle log-likelihood (CWLL) as

$$\begin{aligned} &\ell(\mathbf{y} \mid \mathbf{x}) \\ &= \ell(d_1^1, \ldots, d_1^{n_f}, \ldots, d_{n_s}^1, \ldots, d_{n_s}^{n_f} \mid \mathcal{F}_\mathcal{S}(\mathbf{x})) \\ &= \log(p(d_1^1, \ldots, d_1^{n_f}, \ldots, d_{n_s}^1, \ldots, d_{n_s}^{n_f} \mid \mathcal{F}_\mathcal{S}(\mathbf{x}))), \end{aligned} \quad (13)$$

which models the likelihood of the predicted STFT windows given the STFT windows of the context. The structural constraints of completeness and decomposability of the circuits still hold [Yu et al., 2021a].

Whittle Einsum Network (WEin) is our adaptation of Einsum networks [Peharz et al., 2020] for modeling complex values, which is better suited for the spectral domain. We explore Einsum networks since they are a recent efficient implementation of probabilistic circuits. For time series $\mathbf{x}$, WEin models the Fourier coefficients d_τ^k at frequency k of the τth window. Thus, WEin models the joint distribution

$$p(d_1^1, \ldots, d_1^{n_f}, \ldots, d_{n_s}^1, \ldots, d_{n_s}^{n_f}). \quad (14)$$

Therefore, the Whittle log-likelihood (WLL) is defined as:

$$\begin{aligned} \ell(\mathbf{x}) &= \ell(\mathcal{F}_\mathcal{S}(\mathbf{x})) \\ &= \log(p(d_1^1, \ldots, d_1^{n_f}, \ldots, d_{n_s}^1, \ldots, d_{n_s}^{n_f})), \end{aligned} \quad (15)$$

which models the joint of all STFT windows of a given time series. Given a joint, it is also natural to access the conditional via marginalization: $P_{Y|X}(Y \mid X) = P_{Y,X}(X,Y)/P_X(X)$, where $P_X(X)$ computes the marginal. Thus, although WEin models the joint, given its inference capabilities, it can compute also such conditionals in a tractable way. Therefore, we can employ it in our architecture as Whittle PC in place of the CWSPN (that is learned in a discriminative fashion). In this case, we employ EM for its weight update, since EM is generally more efficient than SGD for such a circuit. More details on our contributions for WEin e.g. multivariate Gaussian leaves are in Appendix E.

3.4 PREDICTIVE UNCERTAINTY SCORE

Deep neural models do not naturally provide an uncertainty quantification for their predictions. This is fundamental e.g. for anomaly detection or to identify when the model's predictions might be wrong and, thus, make the predictions more trustworthy. Bayesian methods for neural forecasting, e.g. Liang [2005], have usually focused on model uncertainty and, considering their computational cost, for deeper architectures simpler approximations are necessary [Gal and Ghahramani, 2016]. There are alternative non-Bayesian methods that provide confidence intervals [Stankeviciute et al., 2021] or scores [Brando et al., 2018].

Another way is to take into account the extreme values seen at training time. Here we follow this path and use the notion of likelihood ratios to provide a quantification of the predictive uncertainty. We take advantage of the tractable inference of the Whittle PCs to provide a score that expresses the uncertainty of a prediction by relating its likelihood with the highest training sample likelihood that is used as a reference. Thus, the predictive uncertainty is proportional to the distance between the likelihood of a prediction and the observed maximum likelihood, scaled by the difference between the extreme observed likelihoods.

Crucially, the CWLL already allows estimating the likelihood for a predicted window in the spectral domain. This enables e.g. to take insights into how predictive likelihood changes in the time domain. Thus, we can leverage the window function w, to project the likelihood to the time domain at time step n:

$$\lambda_{LR}(n) = \max_k \ell(\mathbf{y}^k \mid \mathbf{x}^k) - w(n)\ell(\mathbf{y}_{\mathrm{pred}}(n) \mid \mathbf{x}), \quad (16)$$

where $\ell(\mathbf{y}_{\mathrm{pred}}(n) \mid \mathbf{x})$ denotes the CWLL of a predicted window at time step n given the context, while every other window of the prediction is marginalized, and $\{\mathbf{x}^k, \mathbf{y}^k\}$ is a pair of context and prediction from the training set. To correctly quantify this projection, it is scaled by the distance between the observed maximum and minimum likelihood, defined as

$$\lambda_{LR}^{max} = \max_k \ell(\mathbf{y}^k \mid \mathbf{x}^k) - \min_k \ell(\mathbf{y}^k \mid \mathbf{x}^k). \quad (17)$$

Thus, by using λ_{LR}^{max} as a normalization factor for λ_{LR}, we can estimate the predictive uncertainty with the following log-likelihood ratio score (LLRS):

$$LLRS(n) = \sqrt{|\lambda_{LR}(n)| / \lambda_{LR}^{max}}. \quad (18)$$

In this manner, a likelihood value that is equally low as the worst training sample likelihood (i.e., $\ell(\mathbf{y}_{\mathrm{pred}} \mid \mathbf{x}) = \min_k \ell(\mathbf{y}^k \mid \mathbf{x}^k)$) results in $LLRS = 1$. On training data, larger likelihoods (i.e., $\ell(\mathbf{y}_{\mathrm{pred}} \mid \mathbf{x}) > \min_k \ell(\mathbf{y}^k \mid \mathbf{x}^k)$) result in scores $LLRS < 1$. Thanks to the flexible inference of Whittle PCs, we have derived a point-wise uncertainty estimation of the predictions, back in the time domain.

4 EXPERIMENTAL EVALUATION

To show the benefits of predictive Whittle networks, we investigate the following research questions.

(Q1) Can the uncertainty estimates derived by LLRS be used to distinguish between "good" and "bad" predictions, making the forecasting more trustworthy?

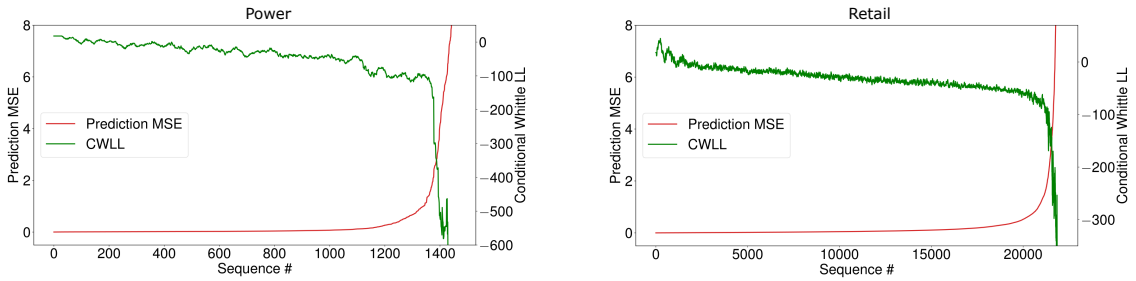

Figure 2: Predictive Whittle networks can correctly separate between "bad" and "good" predictions. This is captured by the correlation of CWLL and MSE on *Power* (Left) and *Retail* (Right). On the x-axis is denoted the enumeration of all test sequences (composed by both context and prediction) in ascending order by MSE. We observe a clear (negative) correlation between a decreasing CWLL and an increasing MSE. The CWLL is smoothed by a moving average of 12 for clarity.

(Q2) By gauging predictive likelihood, can predictive Whittle networks improve the forecasting accuracy, outperforming state-of-the-art forecasters?

The experiments have been run on a GPU NVIDIA GeForce GTX 1070 Ti (8GB VRAM) in a system with CPU Intel i7 4x4,0GHz and 32GB RAM. Our code is publicly available.[1]

4.1 DATA SETS

We evaluate the model performance on three different real-world data sets and apply z-score normalization to normalize the data for all experiments. The first data set is the *Power* consumption from the European Network of Transmission System Operators for Electricity, with a 15-minute sampling rate, available from Wolter et al. [2020]. The task is to predict 1.5 days of power consumption given 14 days of context. Secondly, we investigate the task of predicting the *Retail* demand, using data from a retail location of a big (national) retailer,[2] spanning over 2 years and including roughly 4000 different products with a daily sampling rate. The task is to predict 6 weeks of products demand given a year of context. Furthermore, the well-known *M4* competition data set is employed [Makridakis et al., 2020]. We use window sizes of 96 on *Power* and 24 on *Retail*. Diverse window sizes are applied on *M4* subsets, which are much smaller, making it more challenging for spectral modeling. The step size of STFT is set to half of the window size for each data set. A more detailed description of the data sets, as well as the window sizes on *M4*, are in Appendix F.

4.2 (Q1) USEFUL UNCERTAINTY ESTIMATES

Providing predictive uncertainty in time series forecasting is central. For instance, when performing forecasting in the long run, the prediction error will likely accumulate, leading the model to produce less accurate forecasts.

[1] https://anonymous.4open.science/r/PWN-B7EC/

[2] The name of the company cannot be unveiled due to NDA.

The CWLL provided by Whittle PCs can already be used to distinguish between "bad" and "good" predictions. In particular, a lower CWLL indicates a larger MSE ("bad" prediction), since CWLL negatively correlates with MSE, as visualized in Fig. 2. More specifically, to have a quantitative perspective, by selecting the top 5% sequences with the lowest CWLL from the Whittle PC on *Power*, we find that the 75% of all sequences in the top 5% of highest (i.e. worst) MSEs are included. When looking at the top 10% sequences with the lowest CWLL, 98.5% of all sequences that are in the top 5% of highest (i.e. worst) MSEs are included. Therefore, the likelihood by CWLL can inform the user to distinguish between "good" and "bad" predictions.

Considering that CWLL reflects the prediction quality in the spectral domain, we go one step further, by employing LLRS, which can provide predictive uncertainty estimates back in the time domain, and in turn, indicate when the predictions might be erratic or exceptional. To qualitatively evaluate this ability of predictive Whittle networks with LLRS, we run both standard and long-range prediction on both *Power* and *Retail* data sets. For the *Retail* data set, we predict 8 weeks as standard and 32 weeks as long-range prediction, while the model is trained only for 8 weeks prediction. Similarly, for *Power*, we predict 5 days as standard and 40 days as long-range prediction, with the model trained only for 5 days prediction. Fig. 3 depicts the standard prediction together with the predictive uncertainty score estimated with LLRS. For example, on *Retail*, predictive Whittle networks are able to accurately predict the irregular spike around time step 40, providing low uncertainty scores, while it provides higher uncertainty scores from time step 50 on where the prediction slightly differs from the ground truth. Moreover, as shown in Fig. 4 (Left), the LLRS gives relatively lower scores for predictions from time 2000 to 2700 as the prediction matches the ground truth well, and increases considerably after time 3400, as the predictions diverge from the ground truth. On *Retail*, as shown in Fig. 4 (Right), the more the prediction diverges from the ground truth over longer prediction time, the higher LLRS value we obtain, which indicates the increase of predictive uncer-

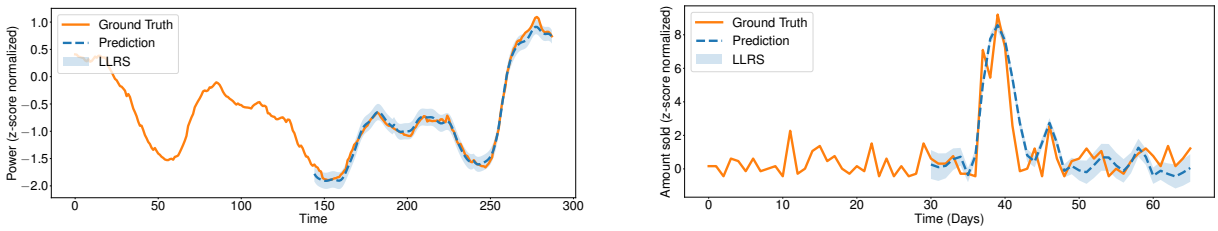

Figure 3: Predictive Whittle networks perform accurate predictions on *Power* (Left) and on a challenging sequence of *Retail* (Right), providing useful predictive uncertainty scores, indicated with LLRS. The context has been cut for clarity.

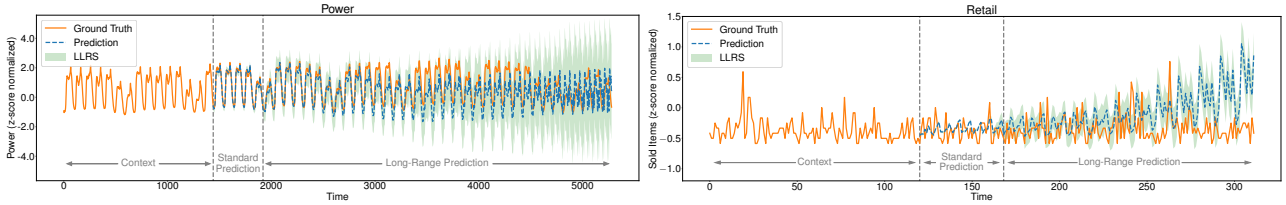

Figure 4: The LLRS from predictive Whittle networks can inform users when the prediction should not be trusted. It increases greatly in the long-range prediction, when prediction is far from the ground truth. The LLRS values at each time step are visualized as bars centered at the corresponding predictions.

tainty. Therefore, the LLRS successfully indicates when the prediction is less trustworthy.

Both CWLL and LLRS can be computed also on an entire sequence, similar analyses can be done in real-world cases, e.g. to detect if a sequence is likely irregular or to sort sequences w.r.t. their CWLL as a surrogate of their expected error when the ground truth is unavailable. With the feedback for the predictions in the time domain, users can have extra knowledge to support decision-making and it is possible to distinguish between potentially "good" and "bad" predictions. We also show additional quantitative analysis by means of a "correlation error" in Appendix G. Therefore, **(Q1)** can be answered affirmatively.

4.3 (Q2) ACCURATE FORECASTING

Setting. We start by comparing predictive Whittle network to its neural spectral forecasters (SRNN and STransformer). In the same spirit, we compare it also against its Whittle PCs (CWSPN and WEin). For the Whittle PCs, preliminary experiments suggested to project the variance to a fixed interval, i.e. $(10^{-4}, 4)$, which corresponds to a standard deviation interval of $(10^{-2}, 2)$. Note that, for a fair comparison, the neural spectral forecasters have similar model capacity to predictive Whittle networks, and in turn, have larger model capacity than PWN's neural components. Besides the real-world data sets *Power* and *Retail* used to show the ability of predictive Whittle networks to provide useful predictive uncertainty estimates, we test its predictive power also on the challenging *M4* where we use sMAPE as loss term in the WFLoss for training and as common evaluation metric [Flores, 1986, Makridakis, 1993].

Furthermore, we compare predictive Whittle networks to several neural forecasters. We start with a simple GRU [Chung et al., 2014], operating in the time domain. Then, we compare with DeepAR [Salinas et al., 2020], as a neural probabilistic competitor which also makes use of additional temporal features. Moreover, we compare to N-Beats, another prominent deep neural architecture. It is composed of different blocks specifically designed for time series forecasting [Oreshkin et al., 2019]. Since predictive Whittle networks do not perform model ensembling, for the comparison, we employ the N-Beats singleton model and use a model configuration similar to the default settings. To have a fair comparison, we provide all models with a capacity similar to the one of the biggest predictive Whittle network variant. See Appendix H for further details. Finally, we also compare with Informer [Zhou et al., 2021], a state-of-the-art attention-based neural forecaster, with its default settings that result in a model with $11.3M$ parameters, i.e. with a capacity at least 11 times larger than predictive Whittle networks. Given its performance, architecture, and model capacity, we use Informer as a gold standard forecaster. We train each model on *Retail*, *Power*, and *M4* for $9k$, $5k$, $15k$ iterations respectively with a batch size of 256, averaging over 5 random seeds.

There exist widely used metrics for probabilistic forecasting, e.g. CRPS [Matheson and Winkler, 1976, Grimit et al., 2006], MSIS [Gneiting and Raftery, 2007] and quantile loss [Koenker and Bassett Jr, 1978]. These are not applicable in our case as they require the probabilities at each time step of the predictions in the time domain that are not obvious to obtain from the spectral domain. Thus, we evaluate on other two common metrics i.e. MSE and sMAPE.

Table 1: Accuracy in MSE for *Retail*, *Power*, and in sMAPE for *M4*, the lower the better. By both operating in the spectral domain and gauging the likelihoods, predictive Whittle networks outperform strong neural forecasters that operate in the time domain, as visible by the **bold**-face best values. Results include standard deviation across five random-seeded experimental repetitions and runner-up performances are also highlighted in bold if they fall within the best value's range.

	MSE		sMAPE on *M4*				
	Power kWh $\cdot 10^5$	Retail Items $\cdot 10^1$	Yearly (23k)	Quarterly (24k)	Monthly (48k)	Others (5k)	Average (100k)
GRU (Time)	23.15 ± 1.87	3.02 ± 0.13	15.54 ± 0.15	11.46 ± 0.05	13.11 ± 0.34	4.97 ± 0.23	12.86 ± 0.22
N-Beats (Time)	4.41 ± 0.12	2.77 ± 0.07	14.17 ± 0.10	**10.98** ± 0.16	12.82 ± 0.21	4.42 ± 0.13	12.27 ± 0.17
DeepAR (Time)	16.83 ± 0.60	2.74 ± 0.02	16.88 ± 0.33	13.26 ± 0.51	14.83 ± 0.32	4.85 ± 0.10	14.43 ± 0.36
Informer (Time)	**3.77** ± 0.09	3.03 ± 0.04	14.49 ± 0.18	11.96 ± 0.43	12.97 ± 0.22	6.33 ± 0.97	12.75 ± 0.30
CWSPN	8.91 ± 1.03	3.57 ± 0.03	23.25 ± 1.95	12.29 ± 0.10	13.82 ± 0.45	9.23 ± 0.13	15.39 ± 0.69
WEin	19.28 ± 0.61	3.72 ± 0.05	39.34 ± 2.28	25.91 ± 1.30	27.07 ± 0.48	12.20 ± 0.48	28.87 ± 1.09
SRNN	4.16 ± 0.06	2.43 ± 0.06	14.25 ± 0.06	11.23 ± 0.06	**12.59** ± 0.04	4.77 ± 0.06	12.26 ± 0.05
STransformer	4.14 ± 0.08	2.70 ± 0.04	15.22 ± 0.51	11.24 ± 0.22	**12.56** ± 0.14	4.67 ± 0.02	12.46 ± 0.24
PWN (SRNN & CWSPN)	4.08 ± 0.08	**2.34** ± 0.03	14.11 ± 0.09	**10.94** ± 0.04	**12.51** ± 0.10	4.58 ± 0.06	**12.11** ± 0.08
PWN (SRNN & WEin)	3.92 ± 0.09	**2.30** ± 0.04	**14.03** ± 0.07	11.28 ± 0.09	**12.54** ± 0.08	4.60 ± 0.03	**12.18** ± 0.08
PWN (STran. & CWSPN)	4.01 ± 0.08	2.66 ± 0.05	15.19 ± 0.27	**10.92** ± 0.16	**12.56** ± 0.09	**4.49** ± 0.06	12.37 ± 0.15
PWN (STran. & WEin)	3.94 ± 0.07	2.68 ± 0.07	15.27 ± 0.28	11.11 ± 0.19	**12.51** ± 0.08	**4.47** ± 0.05	12.41 ± 0.15

Results. Our results are shown in Table 1. Best results of each data set are marked in bold. We can observe that predictive Whittle networks outperform state-of-the-art models in time series forecasting on all data sets except for *Power* where Informer performs best but employing a 11-times larger model capacity, and predictive Whittle networks achieve competitive performance and outperforms all the other baselines that operate in the time domain. Note that STransformer and SRNN do not use any time series-specific component to account for seasonal changes or similar additional temporal features as e.g., N-Beats or DeepAR, but compared to the baselines they still achieve better or competitive accuracy on almost all the cases. Moreover, predictive Whittle networks can take advantage of its two components and exploits the feedback obtained from the predictive likelihoods. In this way, it further improves the results of both its Whittle PC and its neural spectral forecaster.

In general, the variants with WEin as Whittle PC form the best setting for predictive Whittle networks that is also the most parameter-efficient one, having remarkably fewer parameters ($\approx 0.6M$) than competitors (ranging from $0.9M$ to $11M$), details are in Appendix H. Regarding WEin, compared to CWSPN, it has additional advantages since it can answer to a broader set of inference tasks and has a faster convergence [Peharz et al., 2020]. Arguably, the predictions computed by employing only a Whittle PC, obtained via MPE inference (*CWSPN* and *WEin* in Table 1), are generally not as competitive as the ones obtained with neural forecasters. And given its discriminative nature, in this specific task, CWSPN results more accurate than WEin. For a graphical representation of results from Whittle PCs, refer to Appendix I.

In summary, our experimental evidence shows that involving a Whittle PC that provides valuable feedback in form of predictive likelihood to predictive Whittle networks can have a significant impact on time series forecasting. We have shown that, in this way, predictive Whittle networks trained with WFLoss improve accuracy over its individual components and also w.r.t. state-of-the-art neural forecasters, thus, answering **(Q2)** affirmatively.

5 CONCLUSION

We presented predictive Whittle networks with the Whittle forecasting loss as a method to exploit likelihoods to guide the training process towards more accurate spectral forecasting. They outperform state-of-the-art time series forecasters on challenging data sets. Furthermore, thanks to the novel log-likelihood ratio score we introduced, they also provide predictive uncertainty estimates in the time domain based on likelihoods from the spectral domain. This is crucial feedback that can signal users and other systems when a prediction is erratic, making the forecasting more trustworthy. Thus, it can foster users in confident decision-making processes in real-world scenarios. This, in turn, can have several implications on multiple scientific fields where time-series forecasting is of paramount importance. For future work, we envision increased involvement of PCs in hybrid deep neural models to push state-of-the-art on challenging tasks. Moreover, since the Fourier transform can be penalized for short window sizes, improving spectral models on such time series is an interesting future direction.

References

Ahmed Alaa and Mihaela van der Schaar. Attentive state-space modeling of disease progression. In *NeurIPS*, 2019.

Tayfun Alpay, Stefan Heinrich, and Stefan Wermter. Learn-

ing multiple timescales in recurrent neural networks. In *ICANN*, 2016.

Axel Brando, José A. Rodríguez-Serrano, Mauricio Ciprian, Roberto Maestre, and Jordi Vitrià. Uncertainty modelling in deep networks: Forecasting short and noisy series. In *ECML-PKDD*, 2018.

Wessel Bruinsma, Eric Perim, William Tebbutt, J. Scott Hosking, Arno Solin, and Richard E. Turner. Scalable exact inference in multi-output gaussian processes. In *ICML*, 2020.

Kyunghyun Cho, Bart van Merrienboer, Dzmitry Bahdanau, and Yoshua Bengio. On the properties of neural machine translation: Encoder-decoder approaches. In *Workshop on Syntax, Semantics and Structure in Statistical Translation*, 2014.

Junyoung Chung, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. In *NIPS Workshop on Deep Learning*, 2014.

Giorgio Corani, Alessio Benavoli, and Marco Zaffalon. Time series forecasting with gaussian processes needs priors. In *ECML-PKDD*, 2021.

S De Iaco, DE Myers, and D Posa. Strict positive definiteness of a product of covariance functions. *Communications in Statistics-Theory and Methods*, 2011.

Alexiei Dingli and Karl Sant Fournier. Financial time series forecasting–a deep learning approach. *International Journal of Machine Learning and Computing*, 2017.

Benito E Flores. A pragmatic view of accuracy measurement in forecasting. *Omega*, 1986.

Yarin Gal and Zoubin Ghahramani. A theoretically grounded application of dropout in recurrent neural networks. In *NeurIPS*, 2016.

Tilmann Gneiting and Adrian E Raftery. Strictly proper scoring rules, prediction, and estimation. *Journal of the American statistical Association*, 2007.

Eric P Grimit, Tilmann Gneiting, Veronica J Berrocal, and Nicholas A Johnson. The continuous ranked probability score for circular variables and its application to mesoscale forecast ensemble verification. *Quarterly Journal of the Royal Meteorological Society*, 2006.

Chuan Guo, Geoff Pleiss, Yu Sun, and Kilian Q Weinberger. On calibration of modern neural networks. In *ICML*, 2017.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016.

Nicholas J Higham. *Analysis of the Cholesky decomposition of a semi-definite matrix*. Oxford University Press, 1990.

Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 1997.

Agastya Kalra, Abdullah Rashwan, Wei-Shou Hsu, Pascal Poupart, Prashant Doshi, and Georgios Trimponias. Online structure learning for feed-forward and recurrent sum-product networks. In *NeurIPS*, 2018.

Roger Koenker and Gilbert Bassett Jr. Regression quantiles. *Econometrica: journal of the Econometric Society*, 1978.

Binbin Li and Siu-Kui Au. An expectation-maximization algorithm for bayesian operational modal analysis with multiple (possibly close) modes. *Mechanical Systems and Signal Processing*, 2019.

Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In *NeurIPS*, 2019.

Faming Liang. Bayesian neural networks for nonlinear time series forecasting. *Statistics and Computing*, 2005.

Spyros Makridakis. Accuracy measures: theoretical and practical concerns. *International journal of forecasting*, 1993.

Spyros Makridakis, Evangelos Spiliotis, and Vassilios Assimakopoulos. The m4 competition: 100,000 time series and 61 forecasting methods. *International Journal of Forecasting*, 2020.

James E Matheson and Robert L Winkler. Scoring rules for continuous probability distributions. *Management science*, 1976.

Mazen Melibari, Pascal Poupart, Prashant Doshi, and George Trimponias. Dynamic sum product networks for tractable inference on sequence data. In *PGM*, 2016.

Hannes Nickisch and Carl Edward Rasmussen. Approximations for binary gaussian process classification. *Journal of Machine Learning Research*, 2008.

Frank Nielsen and Vincent Garcia. Statistical exponential families: A digest with flash cards. *arXiv preprint arXiv:0911.4863*, 2009.

Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. N-beats: Neural basis expansion analysis for interpretable time series forecasting. In *ICLR*, 2019.

George Papamakarios, Theo Pavlakou, and Iain Murray. Masked autoregressive flow for density estimation. In *NeurIPS*, 2017.

Robert Peharz, Robert Gens, Franz Pernkopf, and Pedro Domingos. On the latent variable interpretation in sum-product networks. *IEEE TPAMI*, 2016.

Robert Peharz, Steven Lang, Antonio Vergari, Karl Stelzner, Alejandro Molina, Martin Trapp, Guy Van den Broeck, Kristian Kersting, and Zoubin Ghahramani. Einsum networks: Fast and scalable learning of tractable probabilistic circuits. In *ICML*, 2020.

Hoifung Poon and Pedro Domingos. Sum-product networks: a new deep architecture. In *UAI*, 2011.

Mohsen Pourahmadi, Michael J Daniels, and Trevor Park. Simultaneous modelling of the cholesky decomposition of several covariance matrices. *Journal of Multivariate Analysis*, 2007.

Syama Sundar Rangapuram, Matthias W Seeger, Jan Gasthaus, Lorenzo Stella, Yuyang Wang, and Tim Januschowski. Deep state space models for time series forecasting. In *NeurIPS*, 2018.

Carl Edward Rasmussen and Christopher K. I. Williams. *Gaussian processes for machine learning*. MIT Press, 2006.

Kashif Rasul, Calvin Seward, Ingmar Schuster, and Roland Vollgraf. Autoregressive denoising diffusion models for multivariate probabilistic time series forecasting. In *ICML*, 2021a.

Kashif Rasul, Abdul-Saboor Sheikh, Ingmar Schuster, Urs M. Bergmann, and Roland Vollgraf. Multivariate probabilistic time series forecasting via conditioned normalizing flows. In *ICLR*, 2021b.

Friedrich Recknagel. Applications of machine learning to ecological modelling. *Ecological modelling*, 2001.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. Learning internal representations by error propagation. Technical report, UCSD, 1985.

David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. Deepar: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting*, 2020.

Masa-aki Sato. Fast learning of on-line em algorithm. *Rapport Technique, ATR Human Information Processing Research Laboratories*, 1999.

Matthias Seeger. Gaussian processes for machine learning. *International journal of neural systems*, 2004.

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 *PGM*, 2020.

Sima Siami-Namini, Neda Tavakoli, and Akbar Siami Namin. A comparison of arima and lstm in forecasting time series. In *ICMLA*, 2018.

Kamile Stankeviciute, Ahmed M Alaa, and Mihaela van der Schaar. Conformal time-series forecasting. In *NeurIPS*, 2021.

K Stankevičiūtė, Ahmed Alaa, and Mihaela van der Schaar. Conformal time-series forecasting. In *NeurIPS*, 2021.

Alexander Y Sun, Dingbao Wang, and Xianli Xu. Monthly streamflow forecasting using gaussian process regression. *Journal of Hydrology*, 2014.

Alex Tank, Nicholas J Foti, and Emily B Fox. Bayesian structure learning for stationary time series. In *UAI*, 2015.

Chiheb Trabelsi, Olexa Bilaniuk, Ying Zhang, Dmitriy Serdyuk, Sandeep Subramanian, Joao Felipe Santos, Soroush Mehri, Negar Rostamzadeh, Yoshua Bengio, and Christopher J Pal. Deep complex networks. In *ICLR*, 2018.

Martin Trapp, Robert Peharz, Franz Pernkopf, and Carl Edward Rasmussen. Deep structured mixtures of gaussian processes. In *AISTATS*, 2020.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NeurIPS*, 2017.

Peter Whittle. The analysis of multiple stationary time series. *Journal of the Royal Statistical Society: Series B (Methodological)*, 1953.

Moritz Wolter and Angela Yao. Complex gated recurrent neural networks. In *NeurIPS*, 2018.

Moritz Wolter, Jürgen Gall, and Angela Yao. Sequence prediction using spectral rnns. In *ICANN*, 2020.

Muqiao Yang, Martin Q Ma, Dongyu Li, Yao-Hung Hubert Tsai, and Ruslan Salakhutdinov. Complex transformer: A framework for modeling complex-valued sequence. In *ICASSP*, 2020.

Zhongjie Yu, Fabrizio Ventola, and Kristian Kersting. Whittle networks: A deep likelihood model for time series. In *ICML*, 2021a.

Zhongjie Yu, Mingye Zhu, Martin Trapp, Arseny Skryagin, and Kristian Kersting. Leveraging probabilistic circuits for nonparametric multi-output regression. In *UAI*, 2021b.

Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *AAAI*, 2021.

APPENDIX

We present supporting material and empirical evidence for our main paper's findings in this appendix. Specifically, the appendix consists of the following sections. We summarize here their content:

- Appendix A: **Training Procedure.** In this section we provide a graphical representation of the predictive Whittle networks training procedure which allows to gauge predictive likelihoods to learn more accurate forecasters in the spectral domain.
- Appendix B: **Short Time Fourier Transform.** In this section we describe the details of the short time Fourier transform and its inverse operation.
- Appendix C: **Improving Spectral RNN.** In this section we describe the details of our SRNN implementation and the preliminary experiments we have conducted to select the best SRNN architecture for predictive Whittle networks.
- Appendix D: **Conceiving the Spectral Transformer (STransformer).** In this section we provide additional details on how we conceived the Spectral Transformer that operates in the complex space and the related experiments.
- Appendix E: **Whittle Einsum Networks (WEin) Implementation.** In this section we describe the implementation of WEin, i.e. our adaptation of Einsum Networks to complex values, better suited to model Fourier transform coefficients.
- Appendix F: **Data Sets.** In this section we describe the data sets we used in our experiments.
- Appendix G: **Correlation Error.** In this section we introduce our method to quantitatively evaluate the quality of the predictive uncertainty estimated by predictive Whittle networks.
- Appendix H: **Experimental Setting and Model Capacity.** Here we provide additional details on the experimental setting and on the capacity of the models employed in the evaluation described in the main paper.
- Appendix I: **Whittle PC Predictions via MPE.** Although not as accurate as neural spectral forecasters, in this section we show that Whittle PCs are able to perform tractable forecasting via MPE inference.

A TRAINING PROCEDURE

As discussed in the main paper, predictive Whittle networks are trained end-to-end in a co-ordinate descent fashion, enabling the Whittle PC to provide feedback to the neural spectral forecaster (denoted as "NSF"). First, Whittle PC weights are optimized (Left), then, NSF weights are optimized by employing the Whittle forecasting loss. The Whittle forecasting loss is based on the NSF predictions as well as the normalized Whittle likelihood ℓ_{norm} obtained from the Whittle PC (Right). These steps are iterated until convergence. A graphical representation of the training procedure is shown in Fig. 5.

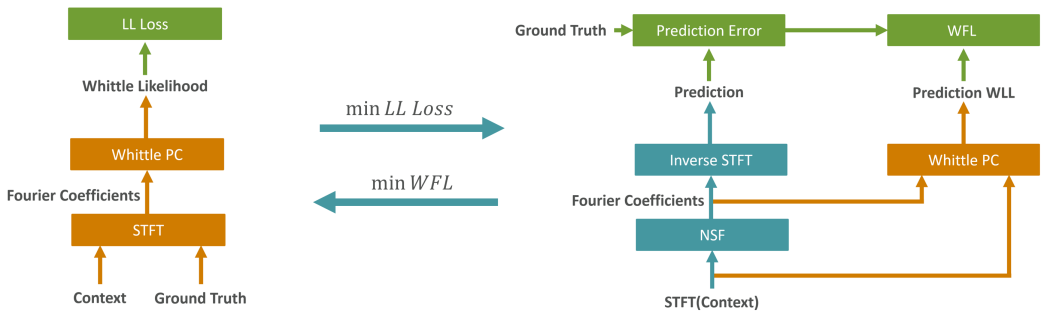

Figure 5: Predictive Whittle networks training procedure together with the Whittle forecasting loss (here denoted as "WFL") allows to gauge the predictive likelihoods provided by the Whittle PC to guide the training towards more accurate neural spectral forecasting.

B SHORT TIME FOURIER TRANSFORM

In the main document, we discuss the benefits of spectral modeling of time series. To obtain a spectral representation of time series, in our work, we employ the short time Fourier transform described in the following, together with its inverse operation.

Given a time series $\mathbf{x} = [x_1, x_2, \cdots x_T]$, denote $\mathbf{x}_\tau^w$ the τth window of $\mathbf{x}$ with width T_w, and $\mathbf{X}_\tau$ the STFT with all frequencies from $\mathbf{x}_\tau^w$. The kth frequency of $\mathbf{X}_\tau$ is denoted as $\mathbf{X}_\tau^k$, and is define as

$$\mathbf{X}_\tau^k = \mathcal{F}_\mathcal{S}(\mathbf{x})_\tau^k = \mathcal{F}_\mathcal{S}(\mathbf{x}_\tau^w)_k = \sum_{t=1}^{T_w} w(S\tau - t) x_t e^{-i\lambda_k t}, \tag{19}$$

where x_t is the tth step in $\mathbf{x}$, $\lambda_k = \frac{2\pi k}{T_w}$. $w(S\tau - t)$ is the truncated Gaussian window function defined as

$$w(n) = \exp(-\frac{1}{2}(\frac{n - T_w/2}{\sigma T_w/2})^2), \tag{20}$$

where n denotes the location of the window and σ is a learnable standard deviation.

Denote $\hat{\mathbf{x}}_\tau^w$ the corresponding inverse short time Fourier transform (iSTFT) of $\mathbf{X}_\tau$, and the tth step of $\hat{\mathbf{x}}_\tau^w$ is defined as

$$\hat{x}_t = {\mathcal{F}_\mathcal{S}}^{-1}(\mathbf{X}_\tau)_t = \frac{\sum_{\tau=-\infty}^{\infty} w(S\tau - t) {\mathcal{F}_t}^{-1}(\mathbf{X}_\tau)}{\sum_{\tau=-\infty}^{\infty} w^2(S\tau - t)}, \tag{21}$$

where $\mathcal{F}_t^{-1}$ is the tth step from the inverse discrete Fourier transform

$$\mathcal{F}_t^{-1}(\mathbf{X}_\tau) = \frac{1}{T_w} \sum_{k=0}^{T_w - 1} \mathbf{X}_\tau^k e^{i\lambda_k t}. \tag{22}$$

C IMPROVING SPECTRAL RNN

With the aim of improving the SRNN presented in Wolter et al. [2020], we run preliminary experiments where we compare four different architectures on the *Power* data set and we examine the impact of our proposals. We test the SRNN as introduced by Wolter et al. [2020], then we add residual connections to it and test performance. Furthermore, we make the model deeper, keep residual connections, add dropout with $p = 0.1$, and test SRNN with two and three layers. To have a fair comparison, we choose the hidden layer sizes for the four aforementioned configurations to be 192, 192, 128, 96 respectively. In this way, each model has approximately the same amount of parameters i.e. $600k$ trainable parameters. Then, we train each model for $4k$ iterations with batch size 256 (80 epochs) with five different seeds and average the results. The results in Table 2 show that residual connections have a remarkably positive impact on the SRNN accuracy (in MSE) making the training also slightly faster. Moreover, the addition of a second layer with dropout results in a further improvement in forecasting (best results in bold). However, a third layer in this setting does not seem to be beneficial empirically. Therefore, for predictive Whittle networks we employ the third architecture i.e. the SRNN with residuals, dropout with $p = 0.1$, and 2 layers.

Table 2: Forecasting accuracy (in MSE) of four different SRNN architecture proposals on the *Power* data set. Results are averaged over five runs with different seeds (best results in bold). Adding residual connections, dropout with $p = 0.1$, and a second layer is beneficial, thus, we will use this architecture for predictive Whittle networks.

	Test MSE [kWh] $\cdot 10^5$	Training time (sec.)
SRNN	4.76 ± 0.076	309 ± 11.0
SRNN + Residuals	4.32 ± 0.063	$\mathbf{299} \pm 10.8$
2 Layers SRNN + Residuals	$\mathbf{4.20} \pm 0.068$	357 ± 13.0
3 Layers SRNN + Residuals	4.22 ± 0.059	415 ± 12.5

As a following step, we run additional experiments to test whether operating with SRNN in the complex space could be beneficial. Thus, we compare our SRNN on *Power* and *Retail* (both data sets are described in the main manuscript) by operating in the real and in the complex space. Table 3 shows that operating in the complex space increases the training times (in seconds) while providing only a marginal improvement in terms of accuracy (in MSE). Therefore, for predictive Whittle networks, we employ the SRNN that operates in the real domain.

Table 3: A comparison of the SRNN operating in the real and in the complex space on *Power* and *Retail* data sets. When operating in the complex space, SRNN requires longer training times (in seconds) while providing only a marginal improvement in terms of accuracy in MSE (best results in bold).

	Power		*Retail*	
	Test MSE [kWh] $\cdot 10^5$	Training time (sec.)	Test MSE [Sold Units] $\cdot 10^1$	Training time (sec.)
SRNN	$\mathbf{4.20} \pm 0.068$	$\mathbf{357} \pm 13.0$	2.45 ± 0.053	$\mathbf{394} \pm 12.2$
Complex SRNN	4.24 ± 0.116	543 ± 24.5	$\mathbf{2.41} \pm 0.097$	593 ± 23.9

D CONCEIVING THE SPECTRAL TRANSFORMER (STRANSFORMER)

In a spectral transformer architecture [Vaswani et al., 2017], analogously to SRNNs, the time steps are considered over n_s windows instead of over the whole sequence. Therefore, it is possible to process long sequences without having to limit the attention size as done e.g. in Yang et al. [2020]. As an example, we consider the *Power* data set (described in the main manuscript). With an input length of 1440, full attention matrices over the input sequences would have a size of 1440^2. In comparison, with our STransformer operating in the spectral domain, the attention matrices would have only size $n_s^2 = 31^2$, which is a drastic reduction in the number of trainable parameters.

We compare the performance on the *Power* data set for three different implementations of STransformer: 1) a (non-complex) STransformer operating in the real space, 2) one that "emulates" the complex space similarly to Yang et al. [2020] 3) our complex STransformer as proposed in the main document. In this way, we can investigate whether complex modeling is beneficial, and we can also examine whether our proposed "native" modeling in the complex space outperforms the "emulated" one. For the non-complex STransformer, we applied the same transformations to the input and the output as described for the SRNN in the main document. All models are equipped with 8 attention heads, a hidden dimension of 64, and a dropout with $p = 0.5$ and have roughly $600k$ parameters. Compared to SRNNs, despite their higher complexity, they need comparable training times thanks to their parallelizability. In these experiments, we train the models for $4k$ iterations with batch size 256 with five different seeds and we average the results. Table 4 indicates that complex modeling is advantageous for transformers (best results in bold). Remarkably, the complex STransformer achieves higher accuracy than the alternatives providing faster training compared to the "emulated" one. While the complex STransformer requires approximately 50% more of the time necessary for the non-complex one, the "emulated" complex STransformer requires about 275% more than the non-complex one. This is due to the increased amount of computations required by the "emulated" complex multi-head attention implementation, which includes eight computations of scaled dot-product attention. Therefore, for predictive Whittle networks, we decide to employ our complex STransformer, since it provides more accurate forecasting with a relatively moderate increase in training time compared to the non-complex STransformer, being also faster and more accurate than the "emulated" one.

Table 4: Preliminary experiments on the *Power* data set show that complex modeling is advantageous for transformer architectures (best results in bold). Compared to non-complex modeling, our complex STransformer improves forecasting accuracy while requiring a moderate amount of additional time for training. The "emulated" complex STransformer is less accurate than the complex STransformer and requires considerable additional time for training.

	Test MSE [kWh] $\cdot 10^5$	Training time (sec.)
STransformer	4.30 ± 0.074	$\mathbf{407} \pm 15.3$
Emulated Complex STransformer	4.25 ± 0.100	1481 ± 41.1
Complex STransformer	$\mathbf{4.16} \pm 0.069$	617 ± 20.5

E WHITTLE EINSUM NETWORKS (WEIN) IMPLEMENTATION

We have introduced WEin in the main body, and here we present the details regarding the extension of the leaf layer with the multivariate Gaussian distribution and its optimization.

E.1 LEAF DISTRIBUTIONS

In EiNets, leaf distributions are represented in the form of exponential families (EFs), for which the log-density of x is given by:

$$\ell(x) = \log h(x) + \mathbb{T}(x)^T \Theta - A(\Theta), \tag{23}$$

where Θ are the natural parameters, $\mathbb{T}$ the sufficient statistics, A the log-normalizer and h the base measure. By means of this representation, one can model several common distributions e.g. Gaussian, Binomial, and Categorial [Peharz et al., 2020]. Furthermore, the representation in expectation form ϕ [Sato, 1999] enables the optimization of the leaf parameters using EM on an abstract level, thus, being independent of the actual employed leaf distribution.

In order to model the covariance matrix $\Sigma_{\mathbf{X}_\tau^k} \in \mathbb{R}^{2\times 2}$ as described in Yu et al. [2021a], we employ a multivariate Gaussian whose EF-form parameters are given by Nielsen and Garcia [2009]:

$$\mathbf{\Theta} = \begin{pmatrix} \Theta_1 \\ \Theta_2 \end{pmatrix} = \begin{pmatrix} \Sigma^{-1}\mu \\ -\frac{1}{2}\Sigma^{-1} \end{pmatrix}, \tag{24}$$

$$\mathbb{T}(\mathbf{x}) = \begin{pmatrix} \mathbf{x} \\ \mathbf{x}\mathbf{x}^\top \end{pmatrix}, \tag{25}$$

$$A(\Theta) = \frac{1}{4} tr(\Theta_2^{-1}\Theta_1\Theta_1^T) - \frac{1}{2}\log|\Theta_2| + \frac{\mathcal{D}}{2}\log\pi, \tag{26}$$

$$h(x) = (2\pi)^{-\mathcal{D}/2}, \tag{27}$$

with $tr(\cdot)$ denoting the trace of a matrix and $\mathcal{D}$ the number of dimensions, in our case $\mathcal{D} = 2$.

E.2 LEAF LAYER OPTIMIZATION

For an EiNet modeling $logP(x)$ the optimization of the leaf layer parameters $\boldsymbol{\phi_L}$ with respect to update ϕ_L is given by Peharz et al. [2016]:

$$\phi_L = \frac{\sum_x p_L(x)\mathbb{T}(x)}{\sum_x p_L(x)}, \tag{28}$$

while $p_L(x)$ is retrieved via auto-differentiation:

$$p_L = \frac{\partial \log P}{\partial \log L} = \frac{1}{P}\frac{\partial P}{\partial \log L} = \frac{1}{P}\frac{\partial P}{\partial L}L. \tag{29}$$

As mentioned above, we need to modify Eq. (28) in order to employ a multivariate Gaussian at the leaves. Modeling the covariance $\Sigma_{d_k^m}$ imposes the constraint of positive-definiteness (PD) to $\Sigma_{d_k^m}$ [De Iaco et al., 2011]:

$$z^T \Sigma_{d_k^m} z > 0,\ \forall z \in \mathbb{R}^{\mathcal{D}}, z \neq 0, \tag{30}$$

which also enforces $\Sigma_{d_k^m}$ to be symmetric. To ensure, that this constraint holds during optimization, we do not learn $\Sigma_{d_k^m}$ directly, but rather its Cholesky decomposition via a lower-triangular matrix G. This approach has been used regularly in various applications [Pourahmadi et al., 2007, Li and Au, 2019]. With $\Sigma_{d_k^m} = GG^T$ and $diag(G) > 0$, $\Sigma_{d_k^m}$ is guaranteed to be PD [Higham, 1990]. Furthermore, only $n_G = \mathcal{D} + \mathcal{D}(\mathcal{D}-1)/2$ parameters need to be modeled (instead of $\mathcal{D}^2$). To update G, i.e. $\phi_L^{'\mathcal{D}+1:\mathcal{D}+n_G}$ in the expectation parameters $\phi_L = (\phi_L^1, ..., \phi_L^{\mathcal{D}+n_G})$, we calculate the Cholesky Decomposition $CD(.)$ of the update $\phi_L^{\mathcal{D}+1:\mathcal{D}+\mathcal{D}^2}$:

$$\phi_L' \leftarrow \begin{pmatrix} \phi_L^{1:\mathcal{D}} \\ CD(\phi_L^{\mathcal{D}+1:\mathcal{D}+\mathcal{D}^2} + \lambda I) \end{pmatrix}. \tag{31}$$

In order to apply CD to matrix A, A must be PD. As $\phi_L^{\mathcal{D}+1:\mathcal{D}+\mathcal{D}^2}$ is only guaranteed to be positive-semi-definite (PSD), as we will show below, we add αI with some small $\alpha > 0$, ensuring $\phi_L^{\mathcal{D}+1:\mathcal{D}+n_G} + \alpha I$ to be PD, as the Identity I is PD:

$$\begin{aligned} &z^T(\phi_L^{\mathcal{D}+1:\mathcal{D}+n_G} + \alpha I)z = \\ &z^T\phi_L^{\mathcal{D}+1:\mathcal{D}+n_G} z + \alpha z^T I z > 0,\ \forall z \in \mathbb{R}^D, z \neq 0. \end{aligned} \tag{32}$$

Now we can prove that $\phi_L^{\mathcal{D}+1:\mathcal{D}+D^2}$ is guaranteed to be PSD:

$$z^T \phi_L^{\mathcal{D}+1:\mathcal{D}+n_G} z \geq 0, \ \forall z \in \mathbb{R}^{\mathcal{D}}. \tag{33}$$

Proof. For simplicity, we omit the index $^{\mathcal{D}+1:\mathcal{D}+\mathcal{D}^2}$:

1. Since $z^T \mathbb{T}(x) z = z^T x x^T z = (z^T x)(z^T x)^T = \|z^T x\|_2^2 \geq 0 \ \forall z \in \mathbb{R}^{\mathcal{D}}$, $\mathbb{T}(x)$ is PSD.
2. As $L > 0$, $P > 0$ and $\partial log(x) > 0, \ \forall x > 0$ by definition, we know $\partial log P > 0$ and $\partial log L > 0$, therefore, $p_L(x) > 0$.
3. As multiplication with the scalar $p_L(x)$ does not influence symmetry, we only need to prove Eq. (33) to show that $p_L(x)\mathbb{T}(x)$ is PSD.
4. Since $z^T p_L(x)\mathbb{T}(x) z = p_L(x) z^T \mathbb{T}(x) z$ and $z^T \mathbb{T}(x) z > 0$ as well as $p_L(x) > 0$, we have $z^T p_L(x)\mathbb{T}(x) z > 0$ and, thus, $p_L(x)\mathbb{T}(x)$ is PSD.
5. Given PSD matrices A, B, it can be shown that $A + B$ is always PSD: $z^T A z = z^T A z + z^T B z \geq 0 \ \forall z \in \mathbb{R}^D$. Therefore, also $\sum_x p_L(x)\mathbb{T}(x)$ PSD.
6. Since $\frac{1}{\sum_x p_L(x)}$ is a scalar, we can proceed as in step 4, thus, $\phi_L = \frac{\sum_x p_L(x)\mathbb{T}(x)}{\sum_x p_L(x))^{-1}}$ is PSD.

Finally, as mentioned previously, one can employ a stochastic online version of EM [Sato, 1999]. This requires the full EM update to be replaced by gliding averages:

$$\boldsymbol{\phi}_{\mathbf{L}} \leftarrow (1-\lambda)\boldsymbol{\phi}_{\mathbf{L}} + \lambda \phi'_{\mathrm{L}}, \tag{34}$$

with $\lambda \in [0, 1]$ as step-size parameter. While it does not lead to a guaranteed increase of the training likelihood in each iteration, as full-batch EM, it typically leads to faster learning [Peharz et al., 2020]. As a last step, similarly to what done in Peharz et al. [2020], we project the variance, i.e., the diagonal of $\Sigma_{d_k^m}$, to a fixed variance interval $[\sigma_{min}, \sigma_{max}]$.

F DATA SETS

The first data set is the *Power* consumption from the European Network of Transmission System Operators for Electricity, with a 15-minute sampling rate. We use the crawled version made available by Wolter et al. [2020]. Given 14 days of context, the network has to predict the power load from noon to midnight of the following day (i.e., 1.5 days). We choose a window size of 96, which corresponds to a full day given the 15-minute sampling rate.

Secondly, we investigate the task of forecasting the *Retail* demand, using data from a retail location of a big (national) retailer, spanning over 2 years and including roughly 4000 different products with a daily sampling rate. Here, the task is to predict six weeks of products demand given a year of context. Since there is no sales data available for Sundays, we filter them out, making a window size of 24 a reasonable choice, i.e., spanning 4 weeks of data. Compared to the *Power*, we deliberately use a smaller window size to verify that our approach performs well with different window sizes. Regarding the low-pass filter of STFT, we apply it with a factor of 4 to the *Power* and with a factor of 2 to the *Retail* data.

Third, we test the predictive power of our model on the well-known challenging *M4* data set. It consists of $100,000$ time series of yearly, quarterly, monthly and other (weekly, daily and hourly) data, which are divided into training and test sets. We refer to Makridakis et al. [2020] for more details of the *M4* data set and the *M4* competition. Note that compared with *Power* and *Retail* data sets, the *M4* data set contains time series with a much smaller length of context ($\mathbf{x}$) and future ($\mathbf{y}$). The window sizes for each subset are 6 for yearly, 8 for quarterly, 18 for monthly, 14 for weekly, 14 for daily, and 24 for hourly. Therefore, the window sizes in *M4* become much smaller, which contain fewer frequencies than *Power* and *Retail*, thus, are less advantageous for spectral modeling.

The step size of STFT is set to half of the window size for both data sets.

G CORRELATION ERROR

To support the answer of **(Q1)** in the main body, that predictive Whittle networks can provide useful predictive uncertainty estimates for time series forecasting, we further introduce the correlation error (CE) as a method to obtain a quantitative evaluation of the quality of the predictive uncertainty estimated by predictive Whittle networks. To provide a correlation

error for the nth test sequence, we first calculate a relative prediction error

$$S^n_{Pred} = \sqrt{\frac{SE(\mathbf{y}^n_{Pred}, \mathbf{y}^n_{GT}) - \min_m SE(\mathbf{y}^m_{Pred}, \mathbf{y}^m_{GT})}{\max_m SE(\mathbf{y}^m_{Pred}, \mathbf{y}^m_{GT}) - \min_m SE(\mathbf{y}^m_{Pred}, \mathbf{y}^m_{GT})}}, \tag{35}$$

where SE denotes the squared error between the predicted future $\mathbf{y}_{Pred}$ and the ground truth $\mathbf{y}_{GT}$. Then, given a context $\mathbf{x}$ we calculate a likelihood score:

$$S^n_{\ell} = \sqrt{\frac{\ell(\mathbf{y}^n_{Pred}|\mathbf{x}^n) - \max_m \ell(\mathbf{y}^m_{Pred}|\mathbf{x}^m)}{\min_m \ell(\mathbf{y}^m_{Pred}|\mathbf{x}^m)}}. \tag{36}$$

The square root is employed to take into account the exponential shape of the conditional Whittle log-likelihood (CWLL), see Fig. 2. Given that the MSE reflects the "ground truth" on where a sequence should be placed in the spectrum from "bad" to "good" predictions, we define the correlation error for the CWLL as the quadratic distance of the scores

$$CE^n = (S^n_{Pred} - S^n_{\ell})^2, \tag{37}$$

where $S^n_{Pred}, S^n_{\ell} \in [0, 1]$ by definition and, therefore, $CE^n \in [0, 1]$. In order to better assess this novel score, we provide a random baseline, which draws likelihood scores randomly from a uniform distribution, i.e. $S^n_{\ell_{\text{random}}} \sim \mathbf{U}(0, 1)$.

To evaluate the correlation error, we compare predictive Whittle networks (SRNN) equipped with a CWSPN or, as an alternative, with a Masked Autoregressive Flow (MAF) [Papamakarios et al., 2017], a state-of-the-art neural density estimator. MAF is integrated into the predictive Whittle networks architecture like CWSPN, therefore, it follows the same training objective. We refer to this architecture as *SRNN-MAF*. For each model, we train and report scores by modeling in the spectral domain as well as in the time domain. For CWSPN, modeling the time series in the time domain degenerates to a CSPN [Shao et al., 2020]. Furthermore, we evaluate three different model sizes, *Small, Medium,* and *Large*. The results and the number of trainable parameters are given in Table 5.

Table 5: Test correlation error (lower is better) for different architectures modeling the time series in the time domain (denoted with "*Time*") or in the spectral domain. A lower score indicates a stronger correlation between CWLL and MSE. The results indicate that predictive Whittle networks can distinguish between "good" and "bad" predictions. Besides, modeling in the spectral domain generally outperforms modeling in the time domain w.r.t. the correlation error, in particular for MAF, where it considerably improves parameter efficiency. Furthermore, for smaller model sizes, predictive Whittle networks achieve the best scores, while MAF is better for models with larger capacity.

	Test Correlation Error					
	Power			*Retail*		
	Small	Medium	Large	Small	Medium	Large
PWN-CWSPN	**0.019**	**0.016**	**0.011**	**0.036**	0.035	0.027
PWN-CSPN (Time)	0.023	0.019	0.017	0.042	**0.031**	0.030
SRNN-MAF	0.045	0.026	**0.011**	0.044	0.033	**0.023**
SRNN-MAF (Time)	0.093	0.058	0.051	0.047	0.045	0.029
Random	0.400			0.455		
#Parameters	300k	900k	3M	30K	70K	200K

In general, modeling in the spectral domain is more beneficial than operating in the time domain, while improving also parameter efficiency. This is more prominent for MAF. Furthermore, SRNN-MAF achieves the best scores on larger model sizes. In comparison, predictive Whittle networks are particularly good with reduced model capacity. It is also important to remark that Whittle PCs, like CWSPNs, can naturally answer to a wider range of probabilistic queries than MAF. Additionally, during our experiments, we observed that PWN equipped with CWSPN is also less sensitive to hyperparameter tuning. Overall, the correlation error obtained with the different architectures is relatively low (i.e. good), also on *Retail* which is a more difficult data set. Moreover, all results are much better than the random baseline.

H EXPERIMENTAL SETTING AND MODEL CAPACITY

In this section, we provide further details on the experimental setting of our evaluation described in Section 4.3 of the main document.

We design the simple GRU [Chung et al., 2014], which operates in the time domain, with 2 recurrent layers, an output projection layer as well as 128 hidden units. For it, we provide the similar model capacity of the neural spectral forecasters used in the comparison (SRNN and STransformer) i.e. roughly $900k$ parameters that is also similar to the model size of the biggest predictive Whittle network variant (see text below and Table 6). Similarly, all DeepAR [Salinas et al., 2020] models have around $1M$ parameters.

Regarding N-Beats, it is composed of different blocks specifically designed for time series forecasting [Oreshkin et al., 2019]. Since our architecture does not perform model ensembling, for the comparison, we employ the N-Beats singleton model and use a model configuration similar to its default settings, with one generic, one seasonality, and one trend block and T-degree of 4, 4, and 2 respectively. With three blocks per stack, N-Beats results in approximately $1.1M$ parameters on *Power* and *Retail*. On *M4* the number of parameters ranges from $927k$ to $1M$, since it depends on the time series (e.g. yearly or monthly). While for predictive Whittle networks, it ranges from $582k$ to $939k$ according to the variants employed, where often the best performing variant has remarkably fewer parameters than N-Beats and the other competitors, like Informer with 11M parameters, being more accurate (see Table 1 of the paper). This further demonstrates that our spectral hybrid architecture is also more parameter efficient than models that operate in the time domain.

Since the model capacity of predictive Whittle networks and N-Beats might vary according to the specific set of time series or to the variants employed, we report the model sizes (in thousands of trainable parameters) in Table 6.

Table 6: Model capacity in **thousands** of trainable parameters for each model for N-Beats and predictive Whittle networks.

			M4			*M4 "Others"*		
	Power	Retail	Yearly	Quarterly	Monthly	Weekly	Daily	Hourly
PWN (SRNN & CWSPN)	959	991	777	781	939	780	783	768
PWN (SRNN & WEin)	635	650	620	620	624	620	620	629
PWN (STran. & CWSPN)	921	953	739	743	901	742	745	731
PWN (STran. & WEin)	597	612	582	583	587	582	582	591
N-Beats	1,133	1,093	929	943	988	997	927	1,030

I WHITTLE PC PREDICTIONS VIA MPE

In Table 1 of the main paper, we have also compared the predictive power of the single components of our architecture i.e. the neural spectral forecasters and the Whittle PCs. The latter perform density estimation by learning the joint distribution (pure generative setting as performed by WEin) or the conditional distribution (more discriminative setting as performed by CWSPN). This is a more general task than forecasting. Nevertheless, although not as accurate as neural forecasters, Whittle PCs can provide valuable predictions by means of the most probable explanation query (MPE), given the context $\mathbf{x}$ as partial observation. For this particular use case, CWSPNs are more accurate than WEins. This is motivated by the more discriminative nature of its design and objective i.e. to model the conditional distribution of the target (the future) $\mathbf{y}$ given the context $\mathbf{x}$. As depicted in Fig. 6, Whittle PCs provide good predictions for *Power* while they are less accurate on predicting an irregular pattern such as a spike on *Retail* (around time step 40). Moreover, when employing MPE for predictions, the predictive uncertainty estimated by the log-likelihood ratio score (LLRS) is relatively low since the MPEs achieve a higher likelihood by definition. Thus, this further motivates the need for a hybrid architecture where the two components work in synergy to provide accurate forecasts and useful predictive uncertainty estimates.

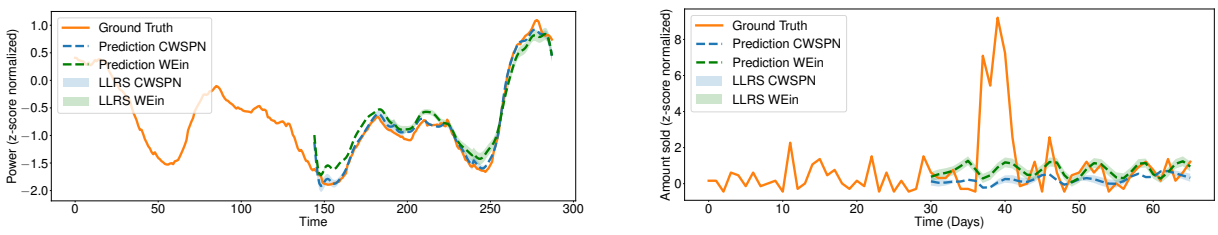

Figure 6: Whittle PCs can also be employed for forecasting via MPE queries. Predictions with the LLRS from CWSPN and WEin on *Power* are on the left and for *Retail* on the right. The context has been cut for clarity. The predictions computed with CWSPN are more accurate given its more discriminative nature.