Predictive Whittle Networks for Time Series Supplementary Material

Zhongjie Yu[*1] **Fabrizio Ventola**[*1] **Nils Thoma**[1]

Devendra Singh Dhami[1,2] **Martin Mundt**[1,2] **Kristian Kersting**[1,2,3]

[1]Department of Computer Science, TU Darmstadt, Darmstadt, Germany
[2]Hessian Center for AI (hessian.AI)
[3]Centre for Cognitive Science, TU Darmstadt

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: **Whittle Likelihood and Whittle Networks.** In this section we describe in more detail the Whittle likelihood and the Whittle networks.
- Appendix B: **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 C: **Short Time Fourier Transform.** In this section we describe the details of the short time Fourier transform and its inverse operation.
- Appendix D: **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 E: **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 F: **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 G: **Data Sets.** In this section we describe the data sets we used in our experiments.
- Appendix H: **Alternative Visualization of LLRS.** In this section we provide an alternative visualization of the LLRS.
- Appendix I: **Correlation Error.** In this section we introduce our method to quantitatively evaluate the quality of the predictive uncertainty estimated by predictive Whittle networks.
- Appendix J: **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 K: **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 WHITTLE LIKELIHOOD AND WHITTLE NETWORKS

The Whittle likelihood models Gaussian stationary multivariate time series in the spectral domain. Following part of the notations in Yu et al. [2021], let $\mathbf{x}_{1:N} = \{\mathbf{x}^1, \dots, \mathbf{x}^N\}$ be N independent realizations of the p dimensional multivariate time series with length T, and $d_{n,k} \in \mathbb{C}^p$ the discrete Fourier coefficient of the n^{th} sequence at frequency $\lambda_k = 2\pi k/T, k = 0, \dots, T-1$:

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

Based on the Whittle approximation assumption [Whittle, 1953], the Fourier coefficients are independent complex normal random variables with mean zero:

$$d_{n,k} \sim \mathcal{N}(0, S_k), \quad k = 0, \dots, T-1, \tag{2}$$

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:

$$S_k = \sum_{h=-\infty}^{\infty} \mathbf{\Gamma}(h) e^{-i\lambda_k h}, \tag{3}$$

[*]Equal Contribution

Accepted for the 38th Conference on Uncertainty in Artificial Intelligence (UAI 2022).

where $\mathbf{\Gamma}(h) = \text{Cov}(x_t, x_{t+h}) \quad \forall t, h \in \mathbb{Z}$. The Whittle likelihood of the N realizations is defined as:

$$p(X_{1:N} \mid S_{0:T-1}) \approx \prod_{n=1}^{N} \prod_{k=0}^{T-1} \frac{1}{\pi^p \left|S_k\right|} e^{-d_{n,k}^* S_k^{-1} d_{n,k}}. \quad (4)$$

While the Whittle approximation holds asymptotically with large T, the Whittle networks relax this approximation by modeling all the Fourier coefficients jointly. With the above relaxation, the Whittle networks are assumed to be able to model both stationary and non-stationary time series.

B 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 by maximizing the likelihood of the context with its ground truth prediction (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. 1. Note that it is also possible to train the Whittle PC with predictions instead of using the ground truth. In this way, one can trade model accuracy with the quality of the predictive uncertainty quantification.

C 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 τ^{th} window of $\mathbf{x}$ with width T_w, and $\mathbf{X}_\tau$ the STFT with all frequencies from $\mathbf{x}_\tau^w$. The k^{th} frequency of $\mathbf{X}_\tau$ is denoted as $\mathbf{X}_\tau^k$, and is define as

$$\mathbf{X}_\tau^k = \mathcal{F}_\mathcal{S}(\mathbf{x})_\tau^k = \mathcal{F}_\mathcal{S}(\mathbf{x}_\tau^w)_k = \sum_{t=1}^{T_w} w(S\tau - t) x_t e^{-i\lambda_k t}, \quad (5)$$

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

$$w(n) = \exp(-\frac{1}{2}(\frac{n - T_w/2}{\sigma T_w/2})^2), \quad (6)$$

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 t^{th} step of $\hat{\mathbf{x}}_\tau^w$ is defined as

$$\hat{x}_t = {\mathcal{F}_\mathcal{S}}^{-1}(\mathbf{X}_\tau)_t = \frac{\sum_{\tau=-\infty}^{\infty} w(S\tau - t)\mathcal{F}_t^{-1}(\mathbf{X}_\tau)}{\sum_{\tau=-\infty}^{\infty} w^2(S\tau - t)}, \quad (7)$$

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

$$\mathcal{F}_t^{-1}(\mathbf{X}_\tau) = \frac{1}{T_w} \sum_{k=0}^{T_w - 1} \mathbf{X}_\tau^k e^{i\lambda_k t}. \quad (8)$$

D 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 1 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.

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 2 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.

E CONCEIVING THE SPECTRAL TRANSFORMER (STRANSFORMER)

In a spectral transformer architecture [Vaswani et al., 2017], analogously to SRNNs, the time steps are considered over

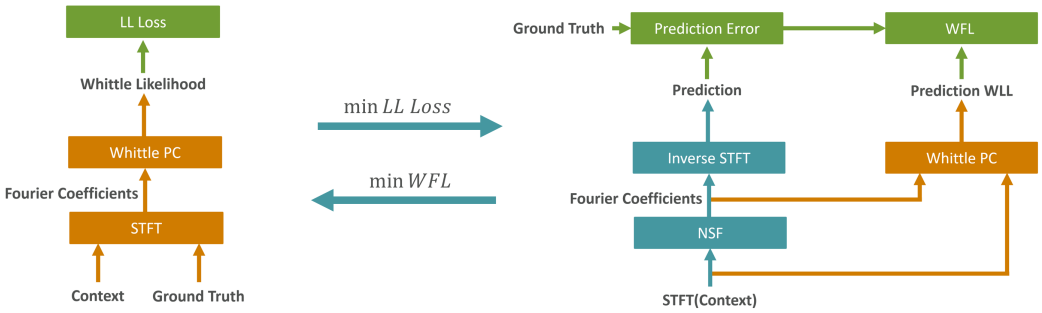

Figure 1: Predictive Whittle networks training procedure together with the Whittle forecasting loss (here denoted as "WFL") allows to gauge the predictive likelihoods provided by the Whittle PC to guide the training towards more accurate neural spectral forecasting.

Table 1: 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

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. 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 3 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.

F 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 multivariate Gaussian distribution and its optimization.

F.1 LEAF DISTRIBUTIONS

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

Table 2: 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

Table 3: 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

given by:

$$\ell(x) = \log h(x) + \mathbb{T}(x)^T \Theta - A(\Theta), \tag{9}$$

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 actually 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. [2021], we employ a multivariate Gaussian whose EF-form parameters are given by Nielsen and Garcia [2009]:

$$\mathbf{\Theta} = \begin{pmatrix} \Theta_1 \\ \Theta_2 \end{pmatrix} = \begin{pmatrix} \Sigma^{-1}\mu \\ -\frac{1}{2}\Sigma^{-1} \end{pmatrix}, \tag{10}$$

$$\mathbb{T}(\mathbf{x}) = \begin{pmatrix} \mathbf{x} \\ \mathbf{x}\mathbf{x}^\top \end{pmatrix}, \tag{11}$$

$$A(\Theta) = \frac{1}{4} tr(\Theta_2^{-1}\Theta_1\Theta_1^T) - \frac{1}{2}\log|\Theta_2| + \frac{\mathcal{D}}{2}\log\pi, \tag{12}$$

$$h(x) = (2\pi)^{-\mathcal{D}/2}, \tag{13}$$

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

F.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]:

$$\phi_L = \frac{\sum_x p_L(x)\mathbb{T}(x)}{\sum_x p_L(x)}, \tag{14}$$

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

$$p_L = \frac{\partial \log P}{\partial \log L} = \frac{1}{P}\frac{\partial P}{\partial \log L} = \frac{1}{P}\frac{\partial P}{\partial L}L. \tag{15}$$

As mentioned above, we need to modify Eq. (14) 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]:

$$z^T \Sigma_{d_k^m} z > 0, \ \forall z \in \mathbb{R}^{\mathcal{D}}, z \neq 0, \tag{16}$$

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 $\phi_L^{\mathcal{D}+1:\mathcal{D}+\mathcal{D}^2}$:

$$\phi_L' \leftarrow \begin{pmatrix} \phi_L^{1:\mathcal{D}} \\ CD(\phi_L^{\mathcal{D}+1:\mathcal{D}+\mathcal{D}^2} + \lambda I) \end{pmatrix}. \tag{17}$$

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:

$$\begin{aligned} &z^T(\phi_L^{\mathcal{D}+1:\mathcal{D}+n_G} + \alpha I)z = \\ &z^T\phi_L^{\mathcal{D}+1:\mathcal{D}+n_G}z + \alpha z^T I z > 0,\ \forall z \in \mathbb{R}^D, z \neq 0. \end{aligned} \tag{18}$$

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

$$z^T\phi_L^{\mathcal{D}+1:\mathcal{D}+n_G}z \geq 0,\ \forall z \in \mathbb{R}^{\mathcal{D}}. \tag{19}$$

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. (19) 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: $z^T Az = z^T Az + z^T Bz \geq 0\ \forall z \in \mathbb{R}^D$. 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:

$$\boldsymbol{\phi}_{\mathbf{L}} \leftarrow (1-\lambda)\boldsymbol{\phi}_{\mathbf{L}} + \lambda\boldsymbol{\phi}'_{\mathrm{L}}, \tag{20}$$

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 $[\sigma_{min}, \sigma_{max}]$.

G 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.

H ALTERNATIVE VISUALIZATION OF LLRS

To provide an alternative visualization of the LLRS in Fig. 4 of the main manuscript, we separate the predictions and LLRS values into two subplots, and stack them vertically for each data set. This is depicted in Fig. 2. In the top plots, we present the predictions together with the ground truth. In the bottom ones, we plot the LLRS scores as curves instead of using bars.

I 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 n^{th} test sequence, we first

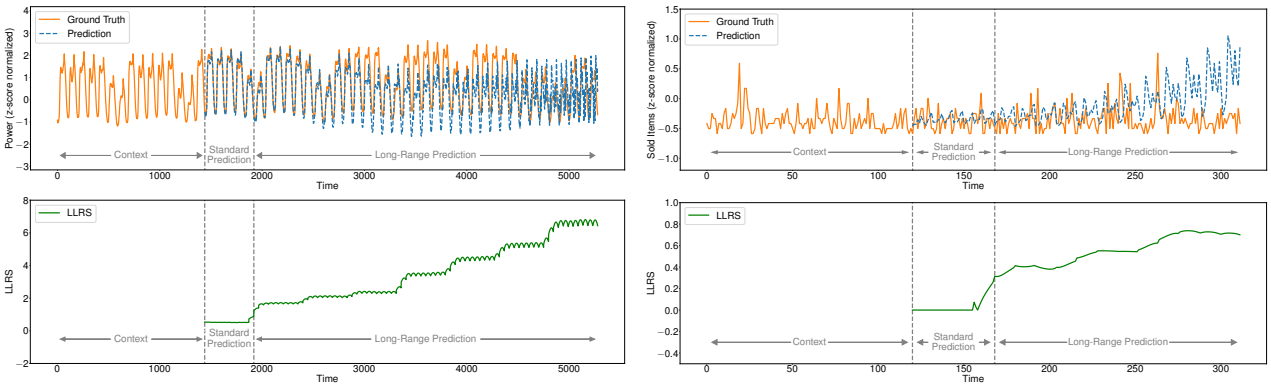

Figure 2: An alternative visualization of the LLRS for long-range predictions on Power and Retail data sets.

calculate a relative prediction error

$$S^n_{Pred} = \sqrt{\frac{SE(\mathbf{y}^n_{Pred}, \mathbf{y}^n_{GT}) - \min_m SE(\mathbf{y}^m_{Pred}, \mathbf{y}^m_{GT})}{\max_m SE(\mathbf{y}^m_{Pred}, \mathbf{y}^m_{GT}) - \min_m SE(\mathbf{y}^m_{Pred}, \mathbf{y}^m_{GT})}}, \tag{21}$$

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:

$$S^n_\ell = \sqrt{\frac{\ell(\mathbf{y}^n_{Pred}|\mathbf{x}^n) - \max_m \ell(\mathbf{y}^m_{Pred}|\mathbf{x}^m)}{\min_m \ell(\mathbf{y}^m_{Pred}|\mathbf{x}^m)}}. \tag{22}$$

The square root is employed to take into account the exponential shape of the conditional Whittle log-likelihood (CWLL), see Fig. 2 in the main paper. 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

$$CE^n = (S^n_{Pred} - S^n_\ell)^2, \tag{23}$$

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 4.

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.

J 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 5). 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

Table 4: 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

Table 5: 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

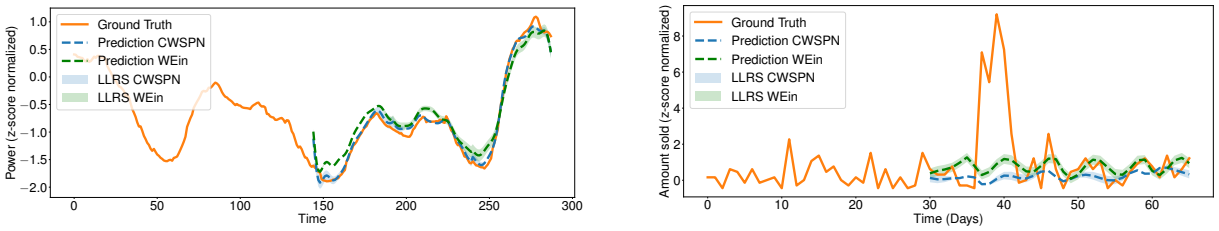

Figure 3: Whittle PCs can also be employed for forecasting via MPE queries. Predictions with the LLRS from CWSPN and WEin on *Power* are on the left and for *Retail* on the right. The context has been cut for clarity. The predictions computed with CWSPN are more accurate given its more discriminative nature.

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 main 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 5.

K 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. 3, 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.