Bridging Self-Attention and Time Series Decomposition for Periodic Forecasting

Song Jiang[1], Tahin Syed[2], Xuan Zhu[2], Joshua Levy[2], Boris Aronchik[2], Yizhou Sun[1]
[1]University of California, Los Angeles, Los Angeles, CA
[2] Amazon Web Services
[1]{songjiang, yzsun}@cs.ucla.edu [2]{tahins, zhuxuan, levyjos, baronchi}@amazon.com

ABSTRACT

In this paper, we study how to capture explicit periodicity to boost the accuracy of deep models in univariate time series forecasting. Recent advanced deep learning models such as recurrent neural networks (RNNs) and transformers have reached new heights in terms of modeling sequential data, such as natural languages, due to their powerful expressiveness. However, real-world time series are often more periodic than general sequential data, while recent studies confirm that standard neural networks are not capable of capturing the periodicity sufficiently because they have no modules that can represent periodicity explicitly. In this paper, we alleviate this challenge by bridging the self-attention network with time series decomposition and propose a novel framework called **DeepFS**. DeepFS equips **Deep** models with **F**ourier **S**eries to preserve the periodicity of time series. Specifically, our model first uses self-attention to encode temporal patterns, from which to predict the periodic and non-periodic components for reconstructing the forecast outputs. The Fourier series is injected as an inductive bias in the periodic component. Capturing periodicity not only boosts the forecasting accuracy but also offers interpretable insights for real-world time series. Extensive empirical analyses on both synthetic and real-world datasets demonstrate the effectiveness of DeepFS. Studies about why and when DeepFS works provide further understanding of our model.

KEYWORDS

time series forecasting, self-attention, time series decomposition, Fourier series

1 INTRODUCTION

Time series is a fundamental data abstract that has diverse real-world use cases, such as product sales [5] and healthcare analytics [6]. With powerful expressiveness for sequential data, neural forecasting models have been introduced into time series with various attempts. For example, DeepAR [22], based on recurrent neural networks (RNNs), outperforms the traditional statistical models by significant margins. Inspired by the success in natural language, transformer architecture [26] has been introduced in modeling time series data recently. [16, 32] refine the vanilla canonical self-attention [26] by exploiting the sparsity of the attention scores. Their customized attention mechanisms lower the high computational cost in modeling long-term time series. [31] expands the embeddings aggregation from point-wise in above works to series-wise. The attention scores are computed over the similar subsequences induced by shifting the original sequence along time.

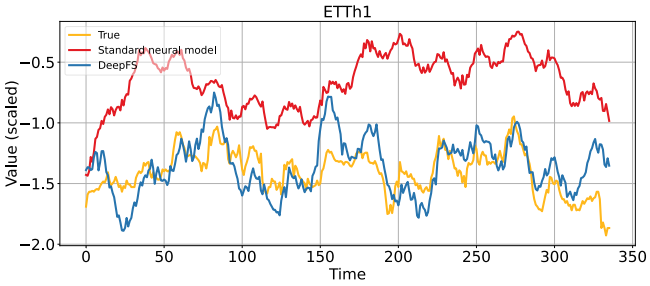

Figure 1: Forecasting v.s. ground-truth on the Electricity Transformer Temperature hourly (ETTh1) dataset [32]. Values are scaled. The standard neural model fails to fully learn the periodicity, aligned with the conclusion in [33]. We propose DeepFS to capture the periodic fluctuations for more accurate real-world time series forecasting.

Although remarkable progress has been made in applying these sequential neural networks on time series, we argue that standard deep learning models are *not* capable of learning the periodicity of time series sufficiently. Fig. 1 shows such a failure example. Recent study [33] reveals that the reason is standard neural nets do not have any modules to capture the periodicity explicitly in their architectures. However, real-world time series typically exhibit stronger periodicity than general sequential data such as audio or text. The inability to learn periodic functions limits the potential of existing neural models for more accurate time series forecasting.

Unlike deep learning, classical time series analysis methods are commonly built on periodicity and other physical-mechanistic factors. A widely used mechanistic approach is time series decomposition [11, 21]. It destructs a time series into seasonality, trend (or trend-cycle), and irregularity, which reflect periodicity, long-term movements and random variation, respectively. However, extrapolating based on these factors alone suffers from poor forecasting accuracy (e.g., ARMIA [11]). This is because the simple decomposition is not fully capable of modeling practical time series where the long-term sequential patterns are complicated.

Contributions: In this paper, inspired by their complementary strengths and weaknesses, we bridge the deep sequential networks and time series decomposition through a simple yet effective

CIKM '22, October 17–21, 2022, Atlanta, GA, USA

ACM ISBN 978-1-4503-9236-5/22/10.
https://doi.org/10.1145/3511808.3557311

encoder-decoder paradigm. The proposed model DeepFS preserves temporal patterns through a self-attention mechanism, from which a periodic inductive bias is employed to capture periodicity for more accurate time series forecasting.

DeepFS is an end-to-end encoder-decoder framework. Composed of the self-attention layers, the encoder converts the historical leading series into latent representations by preserving the inter-dependencies of each timestamp. To capture the periodicity and trend of a sequence, we integrate the time series decomposition in the decoder. Because only an observed sequence can be decomposed, we reformulate the infeasible decomposition of the forecast output as a learning reconstruction problem, namely predicting the periodic and non-periodic components. Specifically, DeepFS uses the Fourier series to represent the periodic component and transforms the leading series embeddings into the parameters of Fourier bases. The learned Fourier series is a periodic inductive bias that explicitly discloses the periodicity of the sequence for better forecasting. Moreover, we use another projection network to generate the prediction of the non-periodic components (i.e., trend) from the leading series embeddings. The final forecasting is an additive combination of these periodic and non-periodic series.

A second benefit of DeepFS is interpretability. In contrast to the lag analysis [31] or score based interpretation (e.g., salience maps [12], attention weights [1] and feature importance [17]), DeepFS learns the mechanistic factors explicitly, including periodicity and trend. For example, we show that DeepFS extracts 24-hour and 12-hour as the periodicity of the wet bulb temperature in Sec. 4. Such factors explicitly explain how the predictions are generated from historical observations and offer fruitful insights into practices such as business planning and decision making.

Empirically, we first justify the periodicity learning module on synthetic datasets. We then present experimental results on four real-world datasets, where DeepFS achieves significant accuracy gains compared to the state-of-the-art transformer models. We also provide analyses of the insightful periods learned from real-world datasets, and extensively study why and when DeepFS can work. We summarize the main contributions of this work as follows:

- We propose DeepFS, a novel model bridging self-attention and time series decomposition for accurate forecasting.
- We propose to inject Fourier series as a learnable periodic inductive bias in DeepFS to capture periodicity.
- DeepFS is also an explainable model that provides insightful interpretations for real-world tasks.
- We conduct comprehensive experiments and extensive analyses on both synthetic and real-world data, demonstrating the effectiveness of DeepFS on time series forecasting.

2 PROBLEM SETUP

In this section, we introduce the univariate time series forecasting problem. Given a L-length observed series $Y' = [Y^{t_0-L+1}, ..., Y^{t_0}] \in \mathbb{R}^L$, where t_0 is the last timestamp with observation, we aim to predict the future H-length sequence $Y = [Y^{t_0+1}, ..., Y^{t_0+H}] \in \mathbb{R}^H$. Following the mechanistic decomposition, we hypothesize that the future time series Y is composed of periodic series (seasonality) $P = [P^{t_0+1}, ..., P^{t_0+H}] \in \mathbb{R}^H$, and non-periodic series (trend) $C = [C^{t_0+1}, ..., C^{t_0+H}] \in \mathbb{R}^H$. Note that both the periodic and non-periodic series are at every timestamp in the future, and are therefore aligned with the forecasting horizon (with same length H). Our goal is to learn a function $f(\cdot)$ that maps the past observations Y' to the periodic series P and non-periodic series C separately, to further reconstruct the future sequences Y. $f(\cdot)$ is formalized as:

$$[Y^{t_0-L+1}, ..., Y^{t_0}] \xrightarrow{f(\cdot)} [P^{t_0+1}, ..., P^{t_0+H}], [C^{t_0+1}, ..., C^{t_0+H}]. \quad (1)$$

Then the final forecasting is an additive combination of P and C, formalized as $[Y^{t_0+1}, ..., Y^{t_0+H}] = [P^{t_0+1}, ..., P^{t_0+H}] + [C^{t_0+1}, ..., C^{t_0+H}]$. We consider capturing the periodicity of future time series Y explicitly for better forecasting. A real-world time series is commonly affected by various causes, e.g., daytime-night, workday-weekend or seasons, and thus its periodicity may consist of multiple sub-components. For example, the product sales typically exhibit weekly and seasonal repeating patterns. We further denote $T = [T^1, ..., T^S] \in \mathbb{R}^S$ the periodicity of time series Y where S is the number of periodic components. Note that the periodic series P can be reconstructed based on the periods T.

3 METHODOLOGY

In this section, we introduce our framework, DeepFS. We argue that a decent design of $f(\cdot)$ in Eq. 1 should follow three principles: (1) simple, with no verbose modules, yet effective; (2) the architecture should be expressive to model the complicated temporal pattern of time series; (3) the periods T should be extracted as much as possible; (4) the prediction should be interpretable for practitioners.

3.1 Overview of DeepFS

Following the above four principles, we depict our proposed model DeepFS in Fig. 2. DeepFS follows the encoder-decoder paradigm, where the encoder coverts the leading values into latent representations at each timestamp with self-attention mechanism, followed by a decoder that transforms these embeddings to the predicted periodic series and the non-periodic series, which reflect periodicity and trend, respectively. The final forecast series is reconstructed by combining the periodic and non-periodic series, following the additive decomposition model [11]. We breakdown these two modules in detail in the following sub-sections.

3.2 Time Series Self-Attention

The RNNs and their variants [10, 20, 22, 23] process the time series data iteratively under the Markov property assumption, i.e., the hidden state encoded at a timestamp is only retained for the next timestamp. Therefore, the impact of a timestamp becomes trivial if the sequence is long, namely RNNs fail to capture long-distance dependency. Hence, we use the self-attention network [26] to model the observed leading time series as it breaks the Markov assumption and is capable of capturing the dependencies between two timestamps even if they are far apart. In this work, we follow the time series self-attention proposed in [32] for time series data, which encodes the value at a time step into latent embeddings via computing its attention strengths with all timestamps in the sequence.

Leading Embeddings Initialization. For later purposes of the attention computation, we first project each numerical value of the leading sequence Y' to a d_u-dimensional vectorized representation

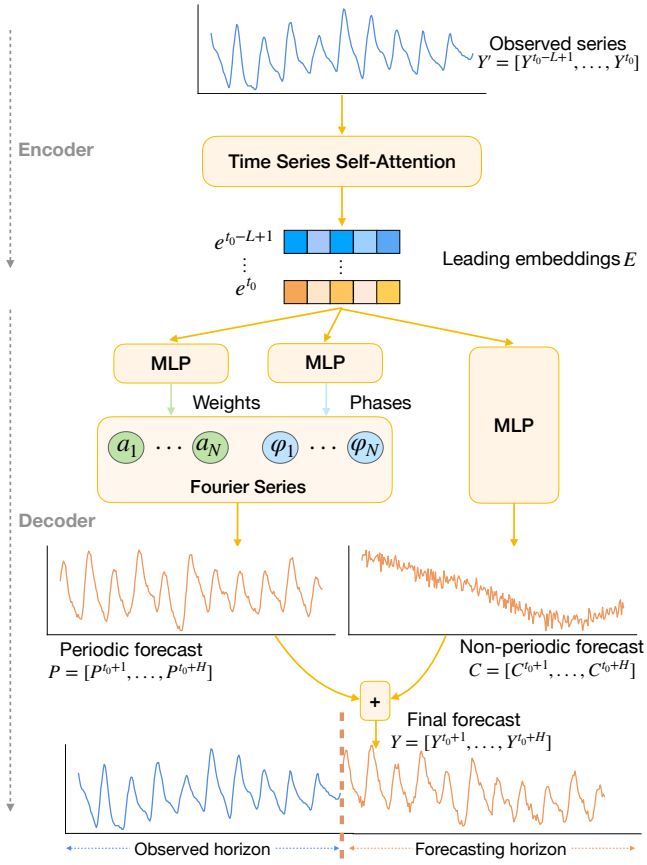

Figure 2: Overview of DeepFS. DeepFS first encodes an observed leading sequence to the leading embeddings E via self-attention. E is then transformed to the parameters of Fourier series (the weights and phases of sinusoidal bases) to predict the periodic series P. Separately, E is also converted to a non-periodic series C that represents the trend prediction. The final time series forecast is an additive combination of the periodic P and non-periodic C.

via a projection operator $\phi(\cdot) : \mathbb{R} \to \mathbb{R}^{d_u}$ as in [32]. The mutual attention mechanism in self-attention inherently discards the relative position contexts, yet the temporal order has proven essential for time series [16, 32]. Therefore, we also include the positional encoding $\tau(\cdot) : \mathbb{R} \to \mathbb{R}^{d_u}$ and temporal encoding $\upsilon(\cdot) : \mathbb{D} \to \mathbb{R}^{d_u}$ used in [32], which inject the local relative orders and the global contexts of timestamps (e.g., hour, week, month) into the sequence representations, respectively. We denote the global contexts of timestamp t by $\tilde{t} \in \mathbb{D}$, where $\mathbb{D}$ is the space of global time[1]. The final initialized leading embeddings $u^t \in \mathbb{R}^{d_u}$ at timestamp t is then formalized as:

$$u^t = \phi(Y^t) + \tau(t) + \upsilon(\tilde{t}). \tag{2}$$

Leading Embeddings with Self-Attention. With the initialized leading embeddings sequence $U = [u^{t_0-L+1}, ..., u^{t_0}]$, we then compute the d_e-dimensional embeddings $E = [e^{t_0-L+1}, ..., e^{t_0}]$ ($e^t \in \mathbb{R}^{d_e}$) for the observed leading sequence Y' that capture its temporal patterns and the inter-dependencies between timestamps via the self-attention mechanism. The self-attention computation is conducted with three matrices: query $Q \in \mathbb{R}^{L \times d_q}$, key $K \in \mathbb{R}^{L \times d_k}$ and value $V \in \mathbb{R}^{L \times d_v}$, which are derived from the initialized embeddings U via three corresponding linear transformations $l_q : \mathbb{R}^{d_u} \to \mathbb{R}^{d_q}$, $l_k : \mathbb{R}^{d_u} \to \mathbb{R}^{d_k}$ and $l_v : \mathbb{R}^{d_u} \to \mathbb{R}^{d_v}$, respectively. d_q, d_k, d_v are the dimensions of the vectors in Q, K and V, where $d_q = d_k$ for the following attention scores computation. The attention based aggregated embeddings $o^t \in \mathbb{R}^{d_v}$ for the leading timestamp t is then formalized as:

$$o^t = \sum_i \frac{\exp(q_t k_i^\top / \sqrt{d_q})}{\sum_j \exp(q_t k_j^\top / \sqrt{d_q})} v_i, \tag{3}$$

where the q_t is the query vector at timestamp t from Q, and i, j are the timestamp indicators of K and V. The final leading embeddings e^t is then transformed from o^t with a further linear projector $l_e : \mathbb{R}^{d_v} \to \mathbb{R}^{d_e}$. The above embeddings aggregation is under the multi-head setting following [26].

With the time series self-attention module, the final leading embeddings $E = [e^{t_0-L+1}, ..., e^{t_0}]$ overcome the long distance challenge of the recursive based temporal aggregation. Note that although we use the attention mechanism from [32], DeepFS is flexible to accommodate other time series transformer methods such as [15, 16, 31].

3.3 Decomposition Based Forecasting

With the leading embeddings E, a straightforward way to model sequential data is using another neural module to convert E into prediction, either a recursive structure [24] or self-attention [32]. Compared to general sequential data such as natural language or audio, we hypothesis that real-world time series are typically more periodic. For example, the traffic volumes exhibit daily and weekly repeating patterns. Intuitively, capturing such periods explicitly has the potential to boost the forecast of future series. However, standard neural nets can not learn the periodicity of time series sufficiently. This is because they do not contain any modules that can represent the periodic functions [33].

To model the periodicity of time series, we take inspirations from the time series decomposition [11]. It deconstructs an observed time series into periodicity, trend and irregularity. We refer to the latter two as non-periodic components. We propose to integrate the time series decomposition in our model, with the mind of capturing periodicity, to generate the forecast series from the leading embeddings E. However, the prerequisite of decomposition is an observed time series, yet the forecast sequences are not available until the model makes final prediction. Therefore, as Fig. 2 shows, we transform the decomposition into a learning based reconstruction problem, i.e., instead of decomposing an observed sequence, we predict the future periodic and non-periodic series based on the leading embeddings E, and then combine them to reconstruct the future series.

Periodic Series Prediction. A periodic sequence can be represented by the Fourier series with appropriate sinusoidal bases, parameterized by weights, periods and phases. Therefore, we inject the Fourier series as a periodic inductive bias in DeepFS to predict

[1]We use "year-month-day-hour-minute-second" for $\mathbb{D}$ in practice.

the periodic series $P = [P^{t_0+1}, ..., P^{t_0+H}]$, which is formalized as:

$$P = a_0 + \sum_{n=1}^{N} a_n \sin(\frac{2\pi nt}{p_0} + \varphi_n), \tag{4}$$

where $t \in [t_0 + 1, ..., t_0 + H]$ (the forecast horizon). N is the number of sinusoidal bases, a_n and φ_n are the weights and phases for the n-th sinusoidal basis, while a_0 is a constant bias term. The periods of the sinusoidal basis set include the basic period p_0 and all its n-th harmonics. Note that to represent an arbitrary periodic sequence, N should be theoretically infinite, which is intractable in practice. Instead, we set N as a tunable parameter and use finite sinusoidal bases to approximate the periodic sequence P. Typically, the periods of the sinusoidal bases set are uniform divisions of forecast horizon H by frequency [19]. Though this set of sinusoidal bases can mimic the periodicity, we argue it can not explicitly represent some meaningful periods by which H are not divisible, e.g., when predicting next month's daily electricity load ($H = 30$), which shows weekly period (7). Therefore, we set the periods of the N bases from 1 to N. A natural choice of N is the forecast horizon H. Our practical experience suggests that if using sliding window to collect training examples, the weights of the sinusoidal bases can still be learned well even if N is larger than H. Therefore, in practices we just set N according to the specific tasks. Another benefit of choosing N regardless of H is that longer period may still be captured even if the forecast horizon H for one data sample is not enough.

As mentioned, the periodic series P can not be derived from post-hoc decomposition, so we turn Eq. 4 into a learnable module in which the sinusoidal bases weights a_n and φ_n are learned from the leading embeddings E with non-linear mapping functions $g_a : \mathbb{R}^{d_e} \to \mathbb{R}^N$ and $g_\varphi : \mathbb{R}^{d_e} \to \mathbb{R}^N$ separately, which are instantiated by the multilayer perceptrons (MLP). This learnable Fourier series serve as the periodic inductive bias to capture the periodicity. Ideally, the weights a_n will be learned to 0 if the sequence does not contain the corresponding period, otherwise some non-trivial values. Therefore, we can infer a sequence's periodicity $T = [T^1, ..., T^S]$ by analyzing the learned weights a_n. It is worth noting that one advantage of using a learnable Fourier series is the explicit periodicity T can be induced, offering human-interpretable insights of the real-world time series compared to just a periodic sequence P as in [19]. We validate this design in Sec. 4.3.

Non-periodic Series Prediction. The non-periodic series $C = [C^{t_0+1}, ..., C^{t_0+H}]$ represents the overall trend of the future sequence. Similar to the periodic series, C is also learnable in our model, which is converted from the leading embeddings E with a non-linear projector $g_c : \mathbb{R}^{d_e} \to \mathbb{R}^H$. A typical approach to the projection from the latent embeddings is the recursive decoding as in seq2seq [24]. Because practical time series may have various trends, here we avoid assuming too many priors and simply use a MLP as g_c in our decoder to generate the forecast of non-periodic series C.

Time series reconstruction. We follow the additive time series decomposition [11] to reconstruct the entire future time series. Namely, the final forecast $Y = [Y^{t_0+1}, ..., Y^{t_0+H}]$ is a summation of the predicted periodic series $P = [P^{t_0+1}, ..., P^{t_0+H}]$ and non-periodic series $C = [C^{t_0+1}, ..., C^{t_0+H}]$. We use the Mean Square Error as the loss function L to compare the forecast and the ground-truth sequences, which is formalized as:

$$L = \frac{1}{H} \sum_{h=1}^{H} (Y^{t_0+h} - Y_g^{t_0+h})^2, \tag{5}$$

where Y_g^t is the ground-truth at timestamp t. The loss L is further averaged over the entire training sequences.

Note that our framework also accommodates for the time series in which the periodicity is insignificant. In this case, all the sinusoidal base weights are learned as some trivial values closed to 0, and DeepFS becomes a standard neural network model for time series. We justify this property empirically in Sec. 4.1.

Interpretaion. A second benefit of DeepFS is the learned periodic and trend series are human-interpretable. Periodicity and trend are mechanistic factors that reveal how a future time series is achieved. In particular, the learned periods T provide practitioners with insights into real-world time series, such as daily repeating patterns of electrical loads. We show the learned periods for various real-world datasets in Sec. 4.2 and Sec. 4.3. Compared to the post-hoc spectral analysis that requires the forecasting series, our periodicity learning is performed up front, and then the learned periods are used to boost prediction accuracy.

4 EXPERIMENTS

In this part, we present the empirical evaluation of DeepFS on univariate time series forecasting. We first justify periodicity learning on synthetic data, then compare DeepFS with diverse baselines on four real-world datasets. We further breakdown DeepFS in various ablation studies to understand why and when DeepFS works.

4.1 Justification on Synthetic Datasets

Since we do not know the ground-truth of either periodicity or the trend on real-world datasets, we first justify periods and non-periodic series learned by DeepFS on synthetic datasets.

Data Synthesis Protocol. We inject all the time series components, i.e., periodicity, trend and randomness, described in [11] in the simulated data. The synthesis protocol is formalized as follows:

$$\sum_{i=1}^{V} a_i^s \sin(\frac{2\pi t}{T_i^s} + \varphi_i^s) + \sum_{j=0}^{W} w_j t^j + \epsilon. \tag{6}$$

The first term represents the periodicity by a combination of V sinusoidal functions, where a_i^s, T_i^s and φ_i^s are the simulated weight, period and phase φ_i^s for the i-th sine wave, respectively. The second term is for the trend. We follow [19] to use a polynomial function with small degree W as the trend of a time series usually does not change severely, and w_j is the weight for j-th power function. The ϵ stands for the randomness in time series synthesis.

In experiments, we select the sinusoidal function number $V \in \{0, 10, 20, 30\}$ to examine how DeepFS works under different periodicity complexity. The periods T_i^s are non-repeatable sampled from the range $[1, 30]$. We set the polynomial function degree W as 2. Other parameters a_i^s, φ_i^s, w_j and ϵ are all randomly generated for each experiment. We simulate 20000 data instances and split them into train/val/test with ratio 0.7/0.1/0.2.

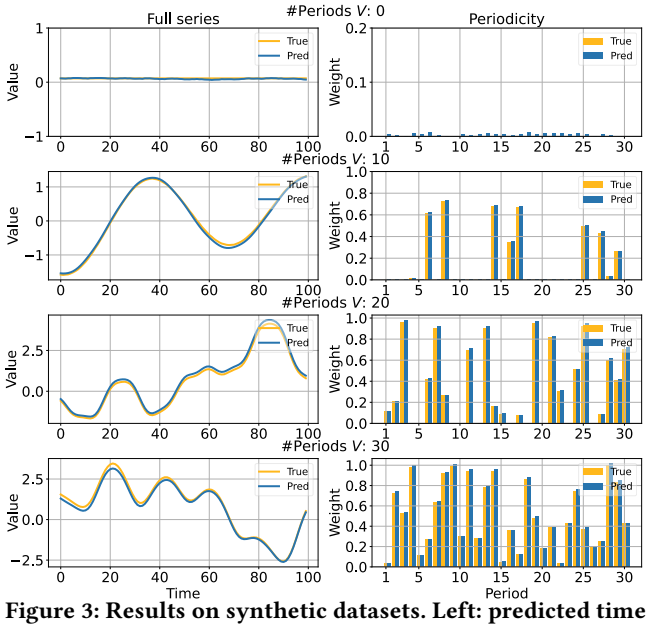

Figure 3: Results on synthetic datasets. Left: predicted time series v.s. ground-truth; Right: predicted periods with their corresponding weights v.s. ground-truth. Rows indicate different numbers of periodicity components V in simulation. The choices of V from top: 0, 10, 20, 30.

Results and analysis. We first report the comparison of the predicted time series and their periodicity with the simulated ground-truth in Fig. 3. Overall, the time series predicted by DeepFS are able to fit the simulated curves closely, while the learned periods weights are also consistent with ground-truth. We notice DeepFS still works even the time series is completely non-periodic (first row in Fig. 3), suggesting the effectiveness of DeepFS on learning true periods rather than just using the sine bases to approximate the sequence. We further summarize the accuracy of predicted periods weights and the non-periodic series in Table. 1, showing that DeepFS could learn both with very low errors. These results demonstrate the correctness of the learned periods and trend, making DeepFS trustworthy to capture periodicity on real-world datasets.

Table 1: Mean absolute error (MAE) of the periods weights and non-periodic series at four periodicity complexities.

#sinusoidals V	Periods weights	Non-periodic series
0	0.0127	0.0315
10	0.0121	0.0325
20	0.0127	0.0582
30	0.0199	0.0555

4.2 Experiments on Real-world Datasets

We then evaluate DeepFS on four real-world datasets to study the accuracy and interpretability of our model in practical scenarios.

Datasets. We use four real-world datasets that are collected by [32]. We describe the datasets as follows:

- **ETTh1, ETTh2, ETTm1:** The ETT (Electricity Transformer Temperature) datasets are metrics that can reflect the electric power deployment. We use 3 ETT datasets ETTh1, ETTh2, ETTm1 released by [32] with granularity 1-hour, 1-hour and 15-mins, respectively. We use the exactly same data split as in [32] (train/val/test: 12/4/4 months).
- **Weather:** The datasets is 4 years of weather records in United States with hourly granularity. We also use the same split as in [32] (train/val/test: 28/10/10 months).

Baselines. We use three-category time series forecasting models as baselines, i.e., (1) statistical model ARIMA [11], (2) RNN-based models LSTMa [2] and DeepAR [22], and (3) transformer based models Reformer [15] and state-of-the-art transformer Informer [32].

Metrics. Because the datasets contain many zeros, we avoid the relative metrics like the mean absolute percentage error (MAPE). Instead, we use the mean square error (MSE) $\frac{1}{D}\frac{1}{H}\sum_{i=1}^{D}\sum_{h=1}^{H}(Y^{t_0+h} - Y_g^{t_0+h})^2$ and mean absolute error (MAE) $\frac{1}{D}\frac{1}{H}\sum_{i=1}^{D}\sum_{h=1}^{H}|Y^{t_0+h} - Y_g^{t_0+h}|$ to compare the DeepFS with baselines, where D is the number of data samples; H is the prediction length; t_0 is the end of leading sequence; Y^t and Y_g^t are the ground-truth and predicted values at timestamp t, respectively.

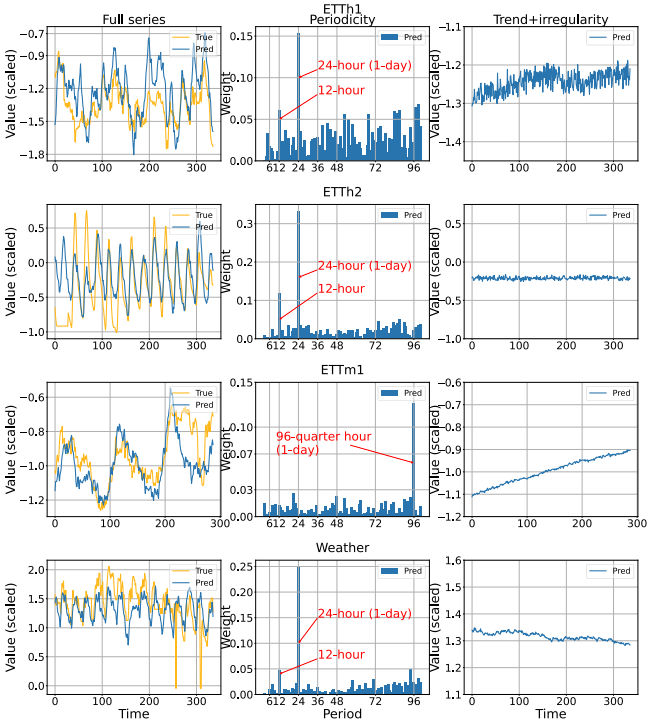

Figure 4: Results on real-world datasets. Left: full predicted time series v.s. ground-truth; Right: predicted trend. Data is scaled. Middle: predicted periods with weights. Striking predicted periods are highlighted with red annotation. From top: ETTh1, ETTh2, ETTm1, Weather.

Implementation Details. We use Pytorch 1.8.1 to conduct our experiments. The initial input embeddings dimension is set to 100. For the encoder, we use 2 self-attention layers with multi-head as 4 and embeddings dimension as 100. We also use layer-normalization and drop-out (0.05) for each self-attention layer. For the decoder, both the MLP modules used for periodic and non-periodic series are 3 layers with hidden embeddings size as 100. We set base number N

Table 2: Accuracy results on ETTh1, ETTh2, ETTm1 and Weather datasets. We use same settings as in [32]. The average MSE and MAE of three runs are reported. Results of all baselines are from [32]. The best model is in boldface for each row. The percentage increases in DeepFS compared to the second-best baseline are listed. "-" symbols refer to worse accuracy.

Dataset	Prediction Length	ARIMA		DeepAR		LSTMa		Reformer		Informer		DeepFS		Gain	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	1d (24)	0.108	0.284	0.107	0.280	0.114	0.272	0.222	0.389	0.092	0.246	**0.090**	**0.234**	2.17%	4.88%
	2d (48)	0.175	0.424	0.162	0.327	0.193	0.358	0.284	0.445	0.158	0.319	**0.121**	**0.274**	23.42%	14.11%
	1w (168)	0.396	0.504	0.239	0.422	0.236	0.392	1.522	1.191	0.183	0.346	**0.130**	**0.284**	28.96%	17.92%
	2w (336)	0.468	0.593	0.445	0.552	0.590	0.698	1.860	1.124	0.215	0.369	**0.098**	**0.246**	54.42%	33.33%
	1m (720)	0.659	0.766	0.658	0.707	0.683	0.768	2.112	1.436	0.257	0.421	**0.167**	**0.329**	35.02%	21.85%
ETTh2	1d (24)	3.554	0.445	0.098	0.263	0.155	0.307	0.263	0.437	**0.093**	**0.240**	0.120	0.270	-	-
	2d (48)	3.190	0.474	0.163	0.341	0.190	0.348	0.458	0.545	0.155	0.314	**0.146**	**0.300**	5.81%	4.46%
	1w (168)	2.800	0.595	0.255	0.414	0.385	0.514	1.029	0.879	0.232	0.389	**0.191**	**0.347**	17.67%	10.69%
	2w (336)	2.753	0.738	0.604	0.607	0.558	0.606	1.668	1.228	0.263	0.417	**0.241**	**0.391**	8.37%	6.24%
	1m (720)	2.878	1.044	0.429	0.580	0.640	0.681	2.030	1.721	**0.277**	0.431	0.281	**0.425**	-	1.39%
ETTm1	6h (24)	0.090	0.206	0.091	0.243	0.121	0.233	0.095	0.228	0.030	0.137	**0.021**	**0.112**	30.00%	18.25%
	12h (48)	0.179	0.306	0.219	0.362	0.305	0.411	0.249	0.390	0.066	0.194	**0.035**	**0.146**	46.97%	24.74%
	1d (96)	0.272	0.399	0.364	0.496	0.287	0.420	0.920	0.767	**0.187**	0.384	**0.187**	**0.345**	0.00%	10.16%
	3d (288)	0.462	0.558	0.948	0.795	0.524	0.584	1.108	1.245	0.401	0.554	**0.219**	**0.386**	45.39%	30.32%
	1w (672)	0.639	0.697	2.437	1.352	1.064	0.873	1.793	1.528	0.512	0.644	**0.248**	**0.416**	51.56%	35.40%
Weather	1d (24)	0.219	0.355	0.128	0.274	0.131	0.254	0.231	0.401	0.117	0.251	**0.102**	**0.231**	12.82%	8.09%
	2d (48)	0.273	0.409	0.203	0.353	0.190	0.334	0.328	0.423	0.178	0.318	**0.139**	**0.272**	21.91%	14.47%
	1w (168)	0.503	0.599	0.293	0.451	0.341	0.448	0.654	0.634	0.266	0.398	**0.216**	**0.350**	18.80%	12.06%
	2w (336)	0.728	0.730	0.585	0.644	0.456	0.554	1.792	1.093	0.297	0.416	**0.280**	**0.413**	5.72%	0.75%
	1m (720)	1.062	0.943	0.499	0.596	0.866	0.809	2.087	1.534	**0.359**	**0.466**	0.394	0.488	-	-

of Fourier series as 100. We ignore the bases with period 1 and 2 because they are easily learned to be constants, causing over-fitting in practice. For the hyperparameters, we use 0.0001 for learning rate, 100 for batch size across all datasets. We use early-stopping based on the error on validation sets to avoid over training, but stopping too early may prevent DeepFS from learning meaningful periods. Therefore, we apply early-stopping after several iterations of training (20 for ETTh1, ETTh2 and Weather; 10 for ETTm1) with 5-step patience in our experiments. We initialize all the model parameters randomly and use Adam [14] as optimizer. All experiments are done within an AWS g4dn.4xlarge instance.

Qualitative results and analysis. We show the forecast curves, the predicted periods and non-periodic series of all datasets in Fig. 4. The learned periodicity for ETTh1 (24-hour, 12-hour), ETTh2 (24-hour, 12-hour), ETTm1 (96-quarter hour), and Weather (24-hour, 12-hour) are all in the daily-wise, which are aligned with human's practical experience of electricity usage and the nature of weather evolution. The learned non-periodic series are also consistent with the overall movements of the ground-truths in general (e.g., a clear increase for ETTm1). Both the prediction of periods and trend contribute to and explain the observation that our predicted time series are able to capture the complicated fluctuations of the ground-truths (left column in Fig. 4), indicating the effectiveness of DeepFS on learning periodicity for better forecasting.

Quantitative results and analysis. We report the forecasting accuracy of DeepFS and baselines in Table. 2. DeepFS outperforms all baselines on 17 out of 20 experiments. Specifically, compared to the state-of-the-art transformer based model Informer, DeepFS improves average 18.4% on MSE and 12.6% on MAE, suggesting the effectiveness of injecting periodic inductive bias into neural networks, as it captures the periodic-trend nature of time series. We notice that Informer is still strong in three cases especially in ETTh2 and Weather datasets. We speculate that the self-attention in Informer's decoder somehow captures the temporal patterns when the periodicity is dominant and clear, such as the ETTh2 and Weather datasets as shown in Fig. 4. However, for more complicated periodic patterns like ETTh1 and ETTm1, our model enjoys non-trivial accuracy gains, especially for long prediction length. We see this as the benefit of explicitly modeling periodicity.

Table 3: Accuracy comparison between DeepFS and variants. DeepFS (P) removes the non-periodic MLP, and DeepFS (NP) removes the Fourier series. The means and standard deviations of five runs are reported. (Numbers) under datasets are prediction length. The best model is in boldface for each row.

Datasets	Metrics	DeepFS (P)	DeepFS (NP)	DeepFS
ETTh1 (336)	MSE	1.990±0.002	0.380±0.175	**0.108±0.007**
	MAE	1.363±0.001	0.529±0.135	**0.262±0.008**
ETTh2 (336)	MSE	1.554±0.002	0.264±0.037	**0.225±0.014**
	MAE	1.118±0.001	0.416±0.033	**0.376±0.013**
ETTm1 (288)	MSE	1.917±0.001	**0.191±0.014**	0.192±0.008
	MAE	1.341±0.000	0.367±0.017	**0.360±0.011**
Weather (336)	MSE	0.989±0.002	0.316±0.012	**0.284±0.009**
	MAE	0.803±0.001	0.439±0.011	**0.420±0.008**

4.3 Why Does DeepFS Work?

We further conduct detailed analyses of DeepFS from five aspects to study the reasons why our model can achieve better accuracy.

Ablation study on model architecture. Our intuition is that both the Fourier series and the non-periodic MLP layers contribute to prediction and are therefore indispensable. To verify this, we compare DeepFS with two variants: only with Fourier series (DeepFS (P)) and only with non-periodic MLP (DeepFS (NP)). We report the accuracy in Table. 3 and draw the predicted series in Fig. 5. We find removing either component leads to a non-trivial accuracy drop,

justifying their necessity. Fig. 5 shows the Fourier series alone is able to capture the fluctuations, further demonstrating the effectiveness of the learnable periodic inductive bias. We notice DeepFS (NP) just learns almost flat lines. We infer it is merely an attempt to reduce the overall loss (e.g., the good numerical error on ETTm1 in Table. 3) rather than preserving the inherent periodic patterns of time series. The results are aligned with the observations in [33] that standard neural networks can not fully learn periodicity.

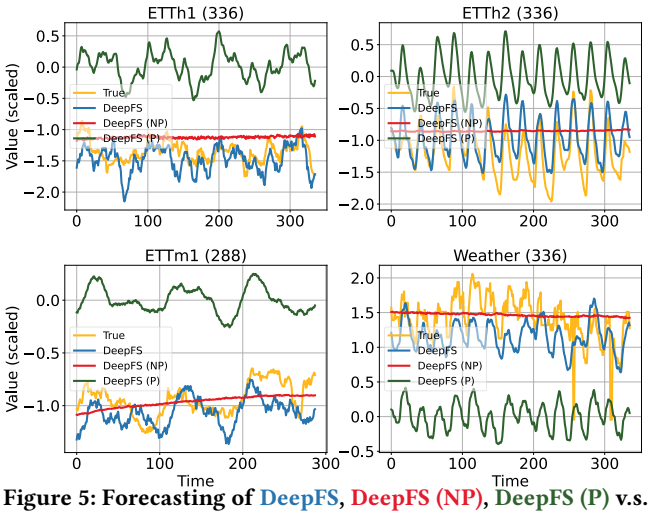

Figure 5: Forecasting of DeepFS, DeepFS (NP), DeepFS (P) v.s. ground-truth on four datasets. From top left: ETTh1, ETTh2, ETTm1, Weather. (Numbers) in titles are prediction lengths. Values are scaled.

Ablation study on sinusoidal bases. The sinusoidal bases are the "silver bullet" to learn periodicity of time series. To measure the impact of the number of sinusoidal bases N, we perform experiments on ETTh2 dataset with 9 different sine bases numbers $N \in \{20, 40, 60, 80, 100, 120, 140, 160, 180\}$. We set the prediction length as 2-week (336 timestamps) and report the accuracy (in MSE and MAE) and predicted periods in Fig. 6. The optimal basis number N in this setting is 40. This is probably because 40 bases (with periods until 40) include the principal periods of ETTh2, i.e., 24-hour and 12-hour. We also observe a rapid drop in accuracy when the bases number is greater than 100, especially 160 and 180. Combining the predicted periods weights, we think the reason is the model fails to learn correct weights for the low-frequency bases (large periods). We infer that an optimal sinusoidal bases number choice should follow two rules: a) must include all principal periodic components, b) when a) holds, use a small number to reduce the difficulty in learning weighting for low-frequency bases. We further explore the failure reason in Sec. 4.4.

Ablation study on leading series length. Our model learns the periodic and non-periodic series from the embeddings of leading sequence. To understand how the length of leading sequence L affect DeepFS, we conduct experiments with 6 different lengths $L \in \{48, 96, 168, 336, 540, 720\}$ on ETTh2 dataset, and report the accuracy and predicted periods in Fig 7. The error first decreases quickly with longer leading sequences, reaches the minimum at L of 96, and then gets worse. However, we note that the learned

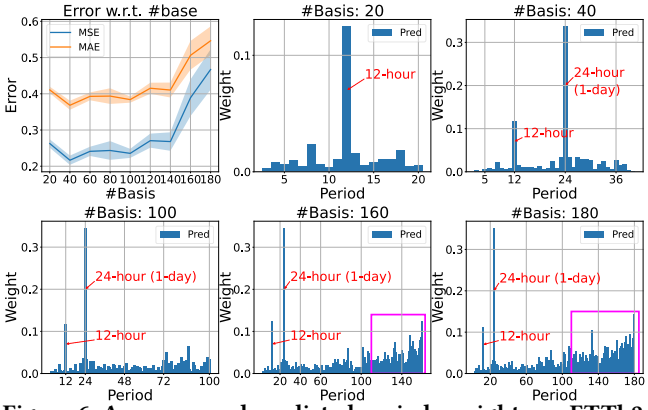

Figure 6: Accuracy and predicted periods weights on ETTh2 dataset with different sinusoidal basis numbers. Solid curves are average of MSE and MAE, ribbons indicate standard derivation. Striking predicted periods are highlighted with red annotation. Magenta rectangles ▭ circle the failures of learning sinusoidal bases weights.

periods are reasonable regardless of leading series lengths. Therefore, we speculate that the non-periodic series patterns are not fully learned with a short leading sequence (pre-96), but the encoder may suffer from the cumbrous attention computation to predict the non-periodic series if the sequence is too long (e.g., post-540).

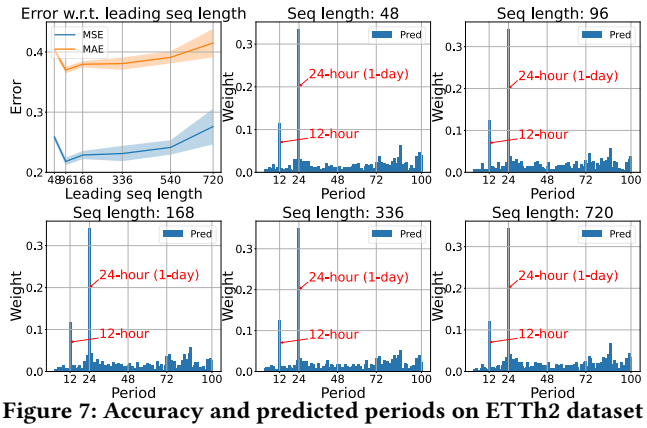

Figure 7: Accuracy and predicted periods on ETTh2 dataset with different leading observed sequence lengths. Solid curves are average of MSE and MAE, ribbons indicate standard derivation. Striking predicted periods are highlighted with red annotation.

Periodicity generalization. Many real-world applications exhibit various periods, e.g., weekly. We further use DeepFS to learn periods for three additional datasets: JHU CSSE COVID-19 Data [8], freeway occupancy rates (traffic)[2], and electricity load [3]. We preprocess all datasets to daily granularity and report the predicted periods in Fig. 8. All three datasets show weekly periods, which are consistent with the practical experience. (We believe the weekly period of COIVD-19 patients is primarily due to the fact that the cases in many states are not fully updated over the weekend.) These

[2]https://pems.dot.ca.gov/
[3]https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014

results confirm that DeepFS is able to capture various granularity periods, indicating the potential use of DeepFS in real scenarios.

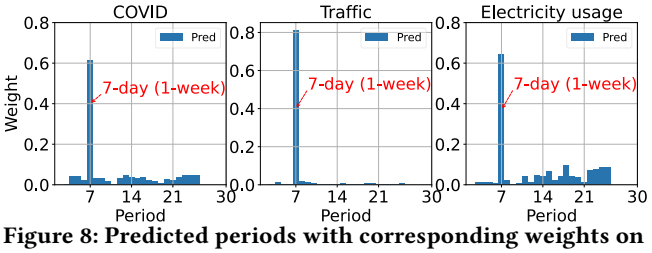

Figure 8: Predicted periods with corresponding weights on daily-granularity datasets. Striking predicted periods are highlighted with red annotation. From left: US COVID-19 death, freeway occupancy rates (traffic), electricity load.

Leading embeddings visualization. To study whether the leading embeddings E are informative for the forecasting, we use t-sne [25] to project the learned leading embeddings E into two-dimensional space in Fig. 9. The 2-D embeddings points exhibit two interesting patterns: (1) continuous from beginning to end of the week (left to right) while parts of weekends are dispersed, and (2) clustered at night-times (bottom left) and during day times (center) separately, with a looping shape for each day. These patterns reflect the periods (e.g., 24-hour) of the leading sequences, reinforcing our conclusion that DeepFS well captures the periodicity.

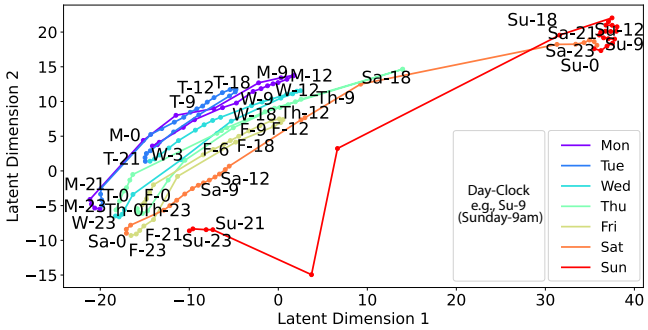

Figure 9: 2D t-SNE projections of leading embeddings E from ETTh2 dataset. Colors are coded by days (Monday to Sunday). "Day-clock" refers to an hour of the day, e.g., "Su-9" is Sunday 9AM. Adjacent times of the same day are connected by lines.

4.4 When Will DeepFS Work?

To further understand when it is appropriate to employ DeepFS for real-world applications, we explore why learning low-frequency sinusoidal bases fails in Sec. 4.3. We test the failure sinusoidal base number $N = 180$ on synthetic data. We compare the predicted periods with different numbers of synthetic data samples in Fig. 10. We find that enough data samples are essential to learning correct weights for low-frequency sinusoidal bases, and thus for low forecast errors. With a fixed length of the leading series, more data samples expand the entire horizon of the observed sequence, which probably explains the success of learning weight for a large period sinusoidal basis. We note that DeepFS may not work for time series with large periods but the available data for training is not sufficient. We leave this "few-shot" problem to future work.

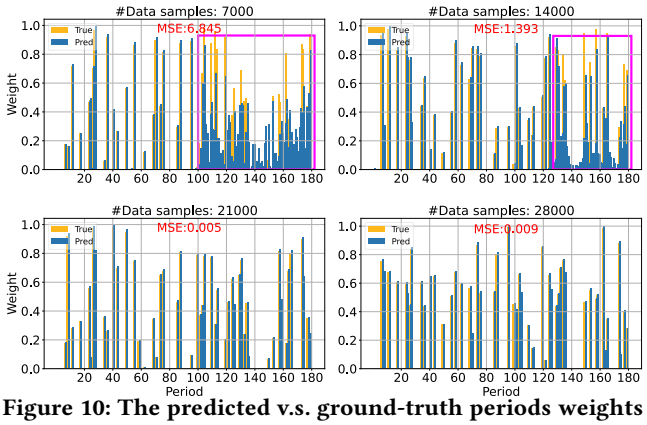

Figure 10: The predicted v.s. ground-truth periods weights with different numbers of synthetic data samples. The periodicity components number $V = 30$ and sinusoidal bases number $N = 180$ for all experiments. The MSE of each experiment is reported. Magenta rectangles ▭ circle the failures of learning sinusoidal bases weights.

5 RELATED WORK

We introduce two lines of related works: general time series forecasting and time series interpretation.

5.1 Time Series Forecasting

Time series forecasting is a ubiquitous problem that has various real-world applications. Traditional forecasting methods focus on modeling the mechanistic factors that cause time series to move. For example, time series decomposition separates the deterministic components of time series from the stochastic parts for extrapolation [7]. The seasonality (i.e., periodicity) and trend are the common factors that are extracted in many decomposition based methods such as ARIMA [21] and the Holt-Winters method [13]. For specific time series tasks, domain knowledge is also used as the mechanistic factors. For example, Wang *et al.* [28] study the citation counts distribution along time and propose models that preserve the citation mechanism as parameters to predict future citation behaviours. The mechanistic factors, either extracted from time series or summarized by domain experts, disclose the evolution of time series and provide intuitive interpretations for how the model makes predictions. However, the traditional mechanistic factors based methods suffer from the poor forecasting accuracy due to their weakness in learning enough insights from the complicated temporal patterns.

Recently, neural networks have been widely applied to predict time series because of their powerful strength to model sequential information. For example, recurrent neural networks (RNNs) based models DeepAR [22] encodes the value at each timestamp with previous hidden representations iteratively, from which to predict the next value by a probabilistic sampler. Convolution based neural networks, such as temporal convolutional network (TCN) [4], are also adapted to time series forecasting, especially for the multivariate setting in which the convolutional layers are used to model the correlations among variables [27]. Both the recurrent and convolutional layers are limited to modeling local time horizons. In contrast, recent transformer based approaches compute the dependencies for every pair of timestamps via the self-attention mechanism [26], which overcomes the long-range challenge of RNNs models. To

improve the efficiency for practical usage, Zhou *et al.* [32] and Li *et al.* [16] exploit the sparsity of self-attention scores and select a subset of pairs to reduce the computational complexity. To model the periodic pattern of time series, Zonoozi *et al* [34] use periodic representations to enhance spatial-temporal forecasting. Fan *et al* [9] and Oreshkin *et al* [19] equip the periodic series as intermediate modules in their deep frameworks. Deep learning approaches have achieved significant progress in time series forecasting, but they have been proved to be unable to fully learn periodicity from time series [33]. In this work, we propose to directly inject the Fourier series as a periodic inductive bias into neural models to capture periodicity explicitly to boost the forecasting accuracy.

5.2 Time Series Interpretation

Another line of related works is the interpretability of time series, which is essential for practitioners to understand how models make predictions and get insights of the real-world task. Scores are a common tool for measuring how a model uses data. Many existing works rely on various scores for explanations. For example, Ismail *et al.* [12] propose a plug-in that first computes the salience scores of timestamps and features separately, and then multiplies them as the measure of importance, which improves the accuracy of locating highly correlated inputs on salience maps. Alaa *et al.* [1] integrate the state transitions with attention modules to predict the disease progression over time where the attention scores are interpreted as the dependency of the disease recovery and the medical contexts. To directly measure the importance of each metadata, Lim *et al.* [17] use weighted combination of metadata during training, the learned weights reflect the contribution of every feature. Wang *et al.* [29] use the partial derivatives to reflect the effects of both the input sequence and the intermediate layers. However, such learned inherent scores are difficult to examine [18] and sometimes even suspicious [3, 30]. Instead, we propose to use the mechanistic factors (i.e., periodicity and trend) that reveal the physical temporal mechanisms to explain the prediction of time series.

6 CONCLUSION

In this paper, we study univariate time series forecasting by explicitly capturing the periodicity. We propose DeepFS, a novel model that combines self-attention and time series decomposition to enhance the periods preserving of neural networks. We achieve this by injecting Fourier series as an periodic inductive bias in our model. Extensive experiments demonstrate that DeepFS achieves better forecasting accuracy. However, DeepFS fails when training data is rare. Future work could study the transferablity of DeepFS for such few-shot scenario. How to learn periodicity for multi-variate time series is also an interesting and useful direction.

ACKNOWLEDGMENTS

The authors sincerely thank all reviewers for their constructive comments and suggestions.

REFERENCES

[1] Ahmed Alaa and Mihaela van der Schaar. 2019. Attentive state-space modeling of disease progression. (2019).
[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. *ICLR*.
[3] Bing Bai, Jian Liang, Guanhua Zhang, Hao Li, Kun Bai, and Fei Wang. 2021. Why Attentions May Not Be Interpretable?. In *Proceedings of the 27th ACM SIGKDD*.
[4] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. 2018. An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. *arXiv preprint arXiv:1803.01271* (2018).
[5] Kasun Bandara, Peibei Shi, Christoph Bergmeir, Hansika Hewamalage, Quoc Tran, and Brian Seaman. 2019. Sales demand forecast in e-commerce using a long short-term memory neural network methodology. In *International conference on neural information processing*. Springer.
[6] Karla L Caballero Barajas and Ram Akella. 2015. Dynamically modeling patient's health state from electronic medical records: A time series approach. In *Proceedings of the 21th ACM SIGKDD*.
[7] Fatoumata Dama and Christine Sinoquet. 2021. Time Series Analysis and Modeling to Forecast: a Survey. *arXiv preprint arXiv:2104.00164* (2021).
[8] Ensheng Dong, Hongru Du, and Lauren Gardner. 2020. An interactive web-based dashboard to track COVID-19 in real time. *The Lancet infectious diseases* (2020).
[9] Wei Fan, Shun Zheng, Xiaohan Yi, Wei Cao, Yanjie Fu, Jiang Bian, and Tie-Yan Liu. 2022. DEPTS: Deep Expansion Learning for Periodic Time Series Forecasting. *arXiv preprint arXiv:2203.07681* (2022).
[10] Jan Gasthaus, Konstantinos Benidis, Yuyang Wang, Syama Sundar Rangapuram, David Salinas, Valentin Flunkert, and Tim Januschowski. 2019. Probabilistic forecasting with spline quantile function RNNs. In *The 22nd international conference on artificial intelligence and statistics*. PMLR.
[11] Rob J Hyndman and George Athanasopoulos. 2018. *Forecasting: principles and practice*. OTexts.
[12] Aya Abdelsalam Ismail, Mohamed K. Gunady, Héctor Corrada Bravo, and Soheil Feizi. 2020. Benchmarking Deep Learning Interpretability in Time Series Predictions. In *Advances in Neural Information Processing Systems*.
[13] Prajakta S Kalekar et al. 2004. Time series forecasting using holt-winters exponential smoothing. *Kanwal Rekhi school of information Technology* (2004).
[14] Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd ICLR*, Yoshua Bengio and Yann LeCun (Eds.).
[15] Nikita Kitaev, Lukasz Kaiser, and Anselm Levskaya. 2020. Reformer: The Efficient Transformer. In *International Conference on Learning Representations*.
[16] Shiyang Li, Xiaoyong Jin, Yao Xuan, Xiyou Zhou, Wenhu Chen, Yu-Xiang Wang, and Xifeng Yan. 2019. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. *NeurIPS* (2019).
[17] Bryan Lim, Sercan Ö Arık, Nicolas Loeff, and Tomas Pfister. 2021. Temporal fusion transformers for interpretable multi-horizon time series forecasting. *International Journal of Forecasting* (2021).
[18] Christoph Molnar. 2019. *Interpretable Machine Learning*.
[19] Boris N Oreshkin, Dmitri Carpov, Nicolas Chapados, and Yoshua Bengio. 2020. N-BEATS: Neural basis expansion analysis for interpretable time series forecasting. In *International Conference on Learning Representations*.
[20] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison Cottrell. 2017. A dual-stage attention-based recurrent neural network for time series prediction. *International Joint Conference on Artificial Intelligence* (2017).
[21] Cleveland Robert, C William, and Terpenning Irma. 1990. STL: A seasonal-trend decomposition procedure based on loess. *Journal of official statistics* 6, 1 (1990).
[22] David Salinas, Valentin Flunkert, Jan Gasthaus, and Tim Januschowski. 2020. DeepAR: Probabilistic forecasting with autoregressive recurrent networks. *International Journal of Forecasting* 36, 3 (2020).
[23] Mike Schuster and Kuldip K Paliwal. 1997. Bidirectional recurrent neural networks. *IEEE transactions on Signal Processing* (1997).
[24] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *NIPS*.
[25] Laurens Van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of machine learning research* 9, 11 (2008).
[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.
[27] Renzhuo Wan, Shuping Mei, Jun Wang, Min Liu, and Fan Yang. 2019. Multivariate temporal convolutional network: A deep neural networks approach for multivariate time series forecasting. *Electronics* (2019).
[28] Dashun Wang, Chaoming Song, and Albert-László Barabási. 2013. Quantifying long-term scientific impact. *Science* 342, 6154 (2013).
[29] Jingyuan Wang, Ze Wang, Jianfeng Li, and Junjie Wu. 2018. Multilevel wavelet decomposition network for interpretable time series analysis. In *Proceedings of the 24th ACM SIGKDD*.
[30] Sarah Wiegreffe and Yuval Pinter. 2019. Attention is not not Explanation. In *EMNLP-IJCNLP*.
[31] Jiehui Xu, Jianmin Wang, Mingsheng Long, et al. 2021. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. *Advances in Neural Information Processing Systems* 34 (2021).
[32] Haoyi Zhou, Shanghang Zhang, Jieqi Peng, Shuai Zhang, Jianxin Li, Hui Xiong, and Wancai Zhang. 2021. Informer: Beyond efficient transformer for long sequence time-series forecasting. In *Proceedings of AAAI*.

[33] Liu Ziyin, Tilman Hartwig, and Masahito Ueda. 2020. Neural networks fail to learn periodic functions and how to fix it. *NeurIPS* (2020).

[34] Ali Zonoozi, Jung-jae Kim, Xiao-Li Li, and Gao Cong. 2018. Periodic-CRN: A convolutional recurrent model for crowd density prediction with recurring periodic patterns.. In *IJCAI*. 3732–3738.