Revitalizing Multivariate Time Series Forecasting: Learnable Decomposition with Inter-Series Dependencies and Intra-Series Variations Modeling

Guoqi Yu [1 2 *] **Jing Zou** [3] **Xiaowei Hu** [4] **Angelica I. Aviles-Rivero** [5] **Jing Qin** [3] **Shujun Wang** [1]

Abstract

Predicting multivariate time series is crucial, demanding precise modeling of intricate patterns, including inter-series dependencies and intra-series variations. Distinctive trend characteristics in each time series pose challenges, and existing methods, relying on basic moving average kernels, may struggle with the non-linear structure and complex trends in real-world data. Given that, we introduce a learnable decomposition strategy to capture dynamic trend information more reasonably. Additionally, we propose a dual attention module tailored to capture inter-series dependencies and intra-series variations simultaneously for better time series forecasting, which is implemented by channel-wise self-attention and autoregressive self-attention. To evaluate the effectiveness of our method, we conducted experiments across eight open-source datasets and compared it with the state-of-the-art methods. Through the comparison results, our **Leddam** (**LE**arnable **D**ecomposition and **D**ual **A**ttention **M**odule) not only demonstrates significant advancements in predictive performance but also the proposed decomposition strategy can be plugged into other methods with a large performance-boosting, from 11.87% to 48.56% MSE error degradation. Code is available at this link: https://github.com/Levi-Ackman/Leddam.

[*] Work done as an intern in PolyU. [1]Department of Biomedical Engineering, The Hong Kong Polytechnic University, Hong Kong SAR, China [2]University of Electronic Science and Technology of China, Sichuan, China [3]School of Nursing, The Hong Kong Polytechnic University, Hong Kong SAR, China [4]Shanghai Artificial Intelligence Laboratory, Shanghai, China [5]DAMTP, University of Cambridge, Cambridge, UK. Correspondence to: Shujun Wang <shu-jun.wang@polyu.edu.hk>.

Proceedings of the 41st International Conference on Machine Learning, Vienna, Austria. PMLR 235, 2024. Copyright 2024 by the author(s).

1. Introduction

The rising demands in diverse real-world domains have generated an urgent requirement for precise multivariate time series forecasting methodologies, as demonstrated in fields like energy management (Dong et al., 2023; Wu et al., 2024; Yi et al., 2023a), weather forecasting (Anonymous, 2024b;c;a), disease control (Yi et al., 2023b; Liu et al., 2023; Zhou et al., 2022b; Ni et al., 2023), and traffic planning (Rangapuram et al., 2018; Zhao et al., 2017; Shao et al., 2022). The foundation of precise forecasting models lies in effectively identifying and modeling intricate patterns embedded in multivariate time series. Two primary patterns (Figure 1(a)) emerge as inter-series dependencies and intra-series variations (Zhang & Yan, 2023). The former delineates the intricate interplay and correlations among distinct variables, while the latter encapsulates both enduring and ephemeral fluctuations within each specific time series.

However, the time series of each constituent variable in multivariate time series data often displays distinctive variations in its trends. Such discrepancies inherent in raw time series may complicate the modeling of inter-series dependencies (Liu et al., 2024). Furthermore, time series in the real world are continually susceptible to distributional shifts induced by the evolution of their trends—a distinctive attribute that adds complexity to modeling the dynamic intra-series variation patterns within the sequences (Taylor & Letham, 2018; Liu et al., 2022b). Therefore, a robust forecasting method should be capable of addressing the following two challenges: (1) How to precisely unravel patterns within raw time series under the interference of trend components. (2) How to efficiently model inter-series dependencies and intra-series variations.

Certain studies (Wu et al., 2021; Zhou et al., 2022c; Wang et al., 2023; Zeng et al., 2023) aim to address the first challenge by conducting trend-seasonal decomposition of the original time series using Moving Average kernel (MOV). However, such an untrainable procedure along with moving average kernels leads to a lack of robustness. Moreover, the uniform assignment of weights to each data point within the sliding window may impede their ability to discern specific patterns. Such limitation becomes apparent when dealing with intricate time series data (RAW data), particularly those

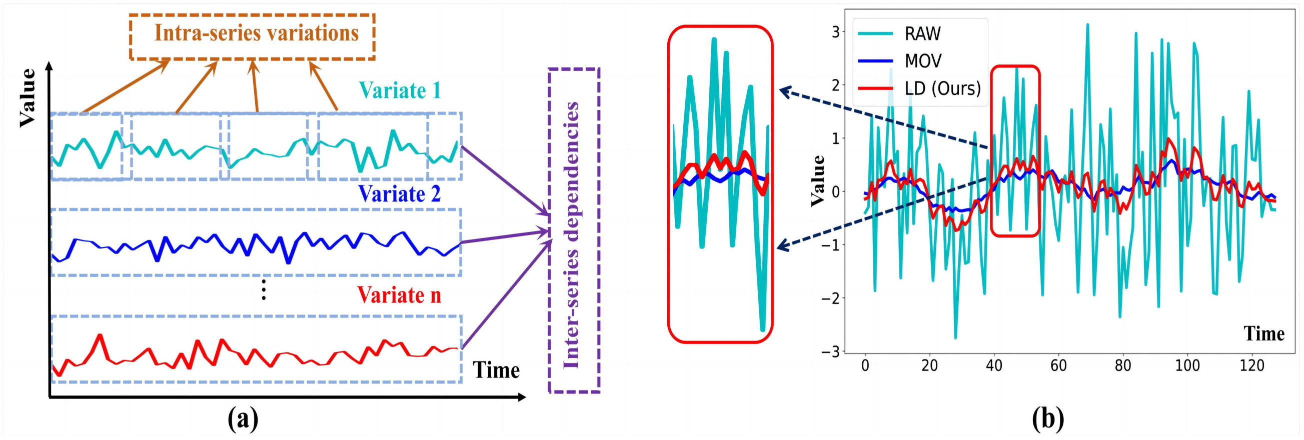


Figure 1. (a) Demonstration of inter-series dependencies and intra-series variations. (b) Visualization of different decomposition schemes in Electricity data. RAW means the raw time series. MOV means moving average kernel, and LD means our learnable decomposition module.

with non-linear structures or significant noise levels, as illustrated in Figure 1(b) (RAW VS. MOV). Therefore, it is necessary to develop a learnable decomposition method to revitalize multivariate time series forecasting tasks.

For the second challenge of modeling inter-series dependencies and intra-series variations in time series forecasting, recent efforts have turned to Transformer architectures renowned for their robust pairwise dependency delineation and multi-level representation extraction within sequences (Vaswani et al., 2017). iTransformer (Liu et al., 2024) effectively handles inter-series dependencies through 'Channel-wise self-attention', embedding entire time series into a token but lacks explicit learning of intra-series variations. Other works directly employ pair-wise attention mechanisms for intra-series variations (Liu et al., 2022a; Zhou et al., 2022a; Li et al., 2019; Liu et al., 2022b). However, they improperly use permutation-invariant attention mechanisms on the temporal dimension (Zeng et al., 2023). Alternatively, some approaches shift to partitioning time series into patches and applying self-attention modeling on these patches (Nie et al., 2023; Zhang & Yan, 2023). Yet, such methods inherently lead to information loss as patches encapsulate only a portion of the original sequence. Moreover, the optimal patch length is also hard to determine. Therefore, we aim to deal with it by generating features suitable for modeling intra-series variations while maximizing information preservation to avoid the above disadvantages.

In this paper, we aim to revitalize multivariate time series forecasting with **Leddam** (**LE**arnable **D**ecomposition and **D**ual **A**ttention **M**odule). Specifically, we first introduce a trainable decomposition module to decompose the original time series data into more reasonable Trend and Seasonal parts. This allows the kernel to prioritize the present data point and adapt to non-linear structures or noise in raw time series, capturing dynamic trend information effectively (see LD in Figure 1(b)). Secondly, we design a 'Dual Attention Module', where 1) channel-wise self-attention to capture inter-series dependencies; 2) an enhanced methodology involving an auto-regressive process and attention mechanism on generated tokens to model intra-series variations. Our **Leddam** aims to provide a more robust and comprehensive solution to time series forecasting challenges. The primary contributions are summarized as follows.

- We propose the incorporation of a learnable convolution kernel initialized with a Gaussian distribution to enhance time series decomposition.
- We devise a 'Dual Attention Module' that adeptly captures both inter-series dependencies and intra-series variations concurrently.
- We validate our **Leddam** by showing that not only demonstrates significant advancements in predictive performance but also the proposed decomposition strategy can be plugged into other methods with a large performance boosting, from **11.87%** to **48.56%** MSE error degradation.

2. Related work

Time Series Data Decomposition. Due to the capacity of the moving average kernel to smooth out short-term fluctuations or noise in the time series, Autoformer (Wu et al., 2021) initially proposed employing the moving average kernel for extracting the trend part of the time series. Later works, including MICN (Wang et al., 2023), FEDformer (Zhou et al., 2022c), DLinear (Zeng et al., 2023), etc., have predominantly adhered to their methodology. However, a rudimentary averaging kernel may inadequately capture precise trends in time series characterized by more intricate patterns than simple linear relationships. It applies uniform weighting to all data points within the window size, which may not be suitable for capturing non-linear or non-stationary trends present in the data.

Intra-series Variations Modeling. It is important to comprehend the temporal variations inherent within the time series for a precise time series forecasting model. Due to the intrinsic limitations of point-to-point attention mechanisms, such as those employed in models like Informer (Zhou et al., 2022a), Reformer (Kitaev et al., 2020), and Pyraformer (Liu et al., 2022a), the resultant attention maps are prone to suboptimality. This arises from the fact that individual points in a time series, in contrast to words (Vaswani et al., 2017) or image patches (Dosovitskiy et al., 2021), lack explicit semantic information. Subsequent endeavours involve partitioning the primary time series into a series of patches (Zhang & Yan, 2023; Nie et al., 2023), followed by applying self-attention mechanisms across these patches to model temporal variations. However, segmenting time series into patches inevitably introduces information loss.

Inter-series Dependencies Modeling. Inter-series dependencies constitute a pivotal attribute that distinguishes multivariate time series from their univariate counterparts. It has come to our attention that the majority of transformer-based methodologies opt to treat values from different variables at the same time step or from distinct channels as tokens (Zhou et al., 2022a; Kitaev et al., 2020; Liu et al., 2022a;b) to model inter-series dependencies. Such strategies may result in attention maps that lack meaningful information, consequently impeding the effective and accurate modeling of the information we seek (Liu et al., 2024). Certain endeavours have sought to adopt channel-independent designs, aiming to mitigate the reduction in predictive accuracy induced by this operation, as exemplified by PatchTST (Nie et al., 2023) and DLinear (Zeng et al., 2023). Channel Independence (CI) regarding variates of time series independently and adopting the shared backbone, have gained consistently increasing popularity in forecasting with performance promotions as an architecture-free method. Recent works (Han et al., 2023; Li et al., 2023) found that while Channel Dependence (CD) benefits from a higher capacity ideally, CI can greatly boost the performance because of sample scarcity, since most of the current forecasting benchmarks are not large enough. However, neglecting the interdependencies among variables may lead to suboptimal outcomes (Liu et al., 2024). Crossformer (Zhang & Yan, 2023) strives to partition time series into patches and subsequently engage in the learning of inter-series dependencies on these patches, a process which, as argued previously, may engender information loss. iTransformer (Liu et al., 2024) achieves more precise modeling of relationships among variables of multivariate time series by embedding the entire time series of a variate into a token, thereby avoiding information loss.

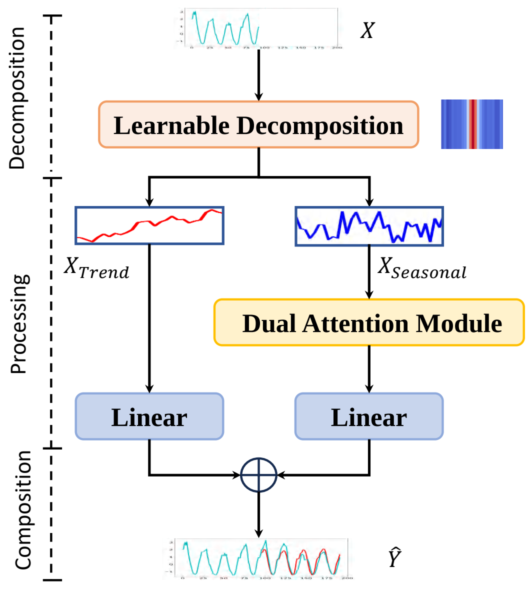


Figure 2. Overall structure of proposed **Leddam**. We start by embedding the time series and incorporating positional encoding. Then, the time series is decomposed into its trend and seasonal parts, each addressed through distinct methodologies. Finally, the processed outcomes of these two components are aggregated to obtain the ultimate predictive result.

3. Methodology

In this section, we elucidate the overall architecture of **Leddam**, which is depicted in Figure 2. We will first define the problem and then describe the proposed Learnable Convolutional Decomposition strategy and Dual Attention Module.

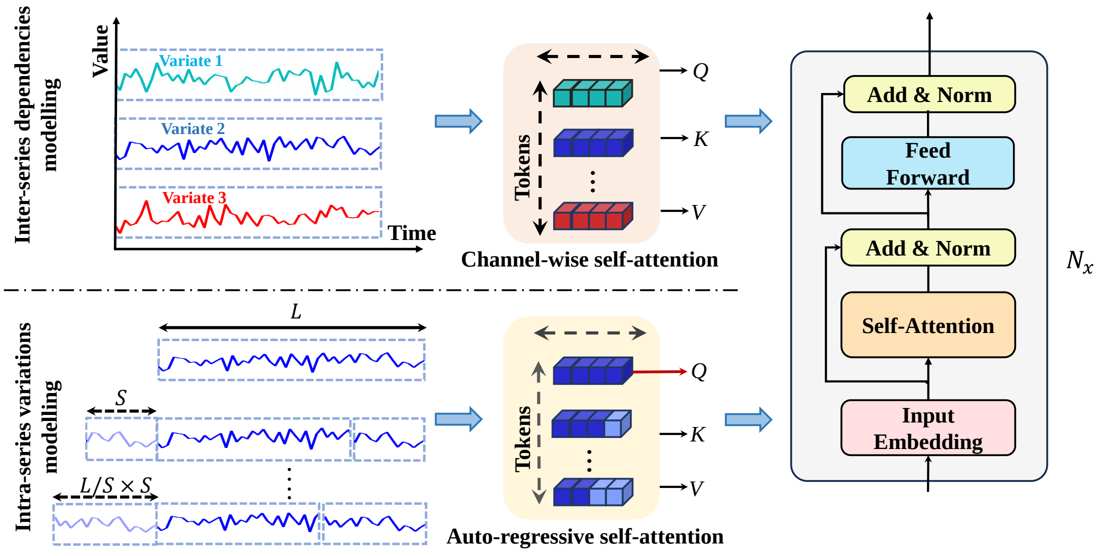
3.1. Problem Definition

Given a multivariate time series input $X \in \mathbb{R}^{N \times T}$, time series forecasting tasks are designed to predict its future F time steps $\hat{Y} \in \mathbb{R}^{N \times F}$, where N is the number of variates or channels, and T represents the look-back window length. We aim to make $\hat{Y}$ closely approximate $Y \in \mathbb{R}^{N \times F}$, which represents the ground truth.

3.2. Learnable Decomposition Module

We employ a superior learnable 1D convolutional decomposition kernel instead of a moving average kernel to comprehensively encapsulate the nuanced temporal variations in the time series.

Projection and position embedding. Following iTransformer (Liu et al., 2024), we first map time series data



Inter-series dependencies modelling
Value
Variate 1
Variate 2
Variate 3
Time
Tokens
Q
K
V
Channel-wise self-attention
Intra-series variations modelling
L
S
L/S × S
Auto-regressive self-attention
Add & Norm
Feed Forward
Add & Norm
Self-Attention
Input Embedding
N_x

Figure 3. General process of 'Dual Attention Module' to deal with Inter-series dependencies and Intra-series variations, respectively. 'Channel-wise self-attention' embeds the whole series of a channel to generate 'Whole Series Embedding', and transformer encoders are employed to model Inter-series dependencies. 'Auto-regressive self-attention' generates 'Auto-regressive Embedding' and still utilizes transformer encoders to model Intra-series variations

$X \in \mathbb{R}^{N\times T}$ from the original space to a new space, subsequently incorporating positional encoding $Pos \in \mathbb{R}^{N\times D}$ as $X_{embed} \in \mathbb{R}^{N\times D}$ following $X_{embed} = (XW + b) + Pos$, with weights $W \in \mathbb{R}^{T\times D}, b \in \mathbb{R}^{1\times D}$, where D is the dimension of the layer.

Learnable 1D convolutional decomposition kernel. To realize the learnable convolutional decomposition, we need to define the convolutional decomposition kernel first. Specifically, we pre-define a stride of $S = 1$ and a kernel size of $K = 25$ experimentally. Regarding its weight, initialization is performed utilizing a Gaussian distribution. We assume its weight is $\omega \in \mathbb{R}^{1\times 1\times K}$, and a hyperparameter $\sigma \in \mathbb{R}$. Here we set $\sigma = 1.0$. Then, we have

$$\begin{gathered} U \in \mathbb{R}^{1\times 1\times K}, \\ U[0,0,i] = \exp\left(-\frac{(i-K/2)^2}{2\sigma^2}\right),\ i = 1,2,\ldots,K, \\ \omega = \mathrm{Softmax}(U, \dim = -1). \end{gathered} \tag{1}$$

Therefore, this initialization results in the central position of the convolutional kernel having the maximum weight, while the edge positions of the kernel have relatively smaller weights. This is typically beneficial for convolutional layers to be more sensitive to the central position when recognizing specific features. Given a multivariate time series input $X_{embed} \in \mathbb{R}^{N\times D}$, where N represents the dimensionality of channels, corresponding to the number of variables in the time series, and D is the dimensionality of the embedding after positional and temporal encoding, to maintain the equivalence of sequence lengths before and after convolution, padding is employed using terminal values. So we get $X_{padded} \in \mathbb{R}^{N\times(D+K-1)}$, we split it into N individual time series $x_i \in \mathbb{R}^{1\times(D+K-1)}, \quad i = 1,2,\ldots,N$. Subsequently, for each x_i, we apply a learnable 1D convolutional kernel with shared weights to extract its trend component denoted as $\hat{x}_i \in \mathbb{R}^{1\times D}$. Subsequently, the convolutional outputs of all $\hat{x}_i$ are concatenated to form the resultant matrix $X_{Trend} \in \mathbb{R}^{N\times D}$. We get the seasonal part $X_{Seasonal} \in \mathbb{R}^{N\times D}$ by $X_{Seasonal} = X_{embed} - X_{Trend}$. The whole process can be summarized as

$$\begin{gathered} X_{Trend} = LD(Padding(X_{embed})), \\ X_{Seasonal} = X_{embed} - X_{Trend}. \end{gathered} \tag{2}$$

Trend part. Given the smoother and more predictable nature of the trend part, we employ a simple MLP for projection to derive the trend part's output akin to (Zeng et al., 2023; Wang et al., 2023), which reads:

$$\begin{gathered} W \in \mathbb{R}^{D\times F}, b \in \mathbb{R}^{1\times F}, \\ X_{T_{out}} = X_{Trend}W + b. \end{gathered} \tag{3}$$

Seasonal part. Considering the suitability of the seasonal component for modeling inter-series dependencies and intra-series variations, we transform $X_{Seasonal} \in \mathbb{R}^{N\times D}$ into two distinct embeddings: 'Whole Series Embedding' and 'Auto-regressive Embedding'. This facilitates the modeling and learning processes for the two patterns.

3.3. Dual Attention Module

We propose a 'Dual Attention Module' to model the inter-series dependencies and intra-series variations simultaneously. Concretely, we devise 'Channel-wise self-attention' for modeling the former and 'Auto-regressive self-attention' for modeling the latter.

Inter-series dependencies modeling. To model inter-series dependencies, follow iTranformer (Liu et al., 2024), we consider $X_{Seasonal}[i,:] \in \mathbb{R}^{1\times D}, \quad i = 1,2,\ldots,N$ as a token as shown in Figure 3. Subsequently, all tokens are sent into a vanilla transformer encoder for learning purposes to get $X_{Inter} \in \mathbb{R}^{N\times D}$:

$$\begin{aligned} Q = X_{Seasonal}W + b, \quad K = X_{Seasonal}W + b, \\ V = X_{Seasonal}W + b, \quad W \in \mathbb{R}^{D\times D}, b \in \mathbb{R}^{1\times D}, \\ Attn = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \\ H = \text{LayerNorm}(X_{Seasonal} + Attn), \\ X_{Inter} = \text{LayerNorm}(FFN(H) + H). \end{aligned} \tag{4}$$

We denote this process as 'Channel-wise self-attention' and the generated embeddings as 'Whole Series Embedding', which, while, maintaining most information of the sequence in contrast to patches or segments, is better suitable for modeling inter-series dependencies, as all semantic information of the variates are saved.

Intra-series variations modeling. To capture intra-series variations, we propose an advanced methodology. Herein, the initial sequence undergoes auto-regressive processing, generating tokens that, while meticulously retaining the entirety of the original information, partially emulate the dynamic variations present in raw time series.

For intra-series variations, as shown in Figure 3, we first split $X_{Seasonal}$ into N individual time series $x_s^i \in \mathbb{R}^{1\times D}, \quad i = 1,2,\ldots,N$. For each x_s^i, given a length L, we generate $S^i \in \mathbb{R}^{\frac{D}{L}\times D}$ tokens by cutting the given length of the sequence from the beginning and concatenate it to the end of the time series:

$$S^i[j,:] = x_s^i[j\cdot L : D] || x_s^i[0 : j\cdot L], \tag{5}$$
$$\text{for } j \in \left\{0,1,\ldots,\left\lfloor\frac{D}{L}\right\rfloor - 1\right\}.$$

Tokens generated in this process are nominated as 'Auto-regressive Embedding'. This auto-regressive token generation approach enables the dynamic simulation of temporal variations within the time series. In contrast to methods involving the utilization of sampling strategies to derive sub-sequences or the partitioning of the primary time series into patches or segments, which inevitably lead to information loss, this approach maximally preserves the information of the original time series. Then we still use another vanilla transformer encoder whose weight is shared across all channels to model intra-series variations, but considering our primary interest in the temporal information of the raw sequence, we designate the raw sequence x_s^i solely as Q, while utilizing the entire sequence S^i as both K and V:

$$\begin{aligned} Q = x_s^i W + b, \quad K = S^i W + b, \quad V = S^i W + b, \\ Attn = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \\ H = \text{LayerNorm}(x_s^i + Attn), \\ X_{Intra}^i = \text{LayerNorm}(FFN(H) + H), \\ X_{Intra} = X_{Intra}^1 || X_{Intra}^2 || \ldots X_{Intra}^n. \end{aligned} \tag{6}$$

And the output X_{Intra}^i of all channels is concatenated as $X_{Intra} \in \mathbb{R}^{N\times D}$. The ultimate output results of the seasonal part have been obtained by:

$$\begin{aligned} W \in \mathbb{R}^{D\times F}, b \in \mathbb{R}^{1\times F}, \\ X_{S_{out}} = (X_{Inter} + X_{Intra})W + b. \end{aligned} \tag{7}$$

Prediction generating. We obtain the ultimate predictive outcomes through $\hat{Y} = X_{S_{out}} + X_{T_{out}}$.

4. Experiments

4.1. Experimental Settings

In this section, we first introduce the whole experiment settings under a fair comparison. Secondly, we illustrate the experiment results by comparing Leddam with the eight current state-of-the-art (SOTA) methods. Further, we conducted an ablation study to comprehensively investigate the effectiveness of the learnable convolutional decomposition module and the effectiveness of the Dual attention module.

Table 1. The Statistics of the eight datasets used in our experiments.

Datasets	ETTh1&2	ETTm1&2	Traffic	Electricity	Solar-Energy	Weather
Variates	7	7	862	321	137	21
Timesteps	17,420	69,680	17,544	26,304	52,560	52,696
Granularity	1 hour	5 min	1 hour	1 hour	10 min	10 min

Datasets. We conduct extensive experiments on selected eight widely-used real-world multivariate time series forecasting datasets, including Electricity Transformer Temperature (ETTh1, ETTh2, ETTm1, and ETTm2), Electricity, Traffic, Weather used by Autoformer (Wu et al., 2021), and Solar-Energy datasets proposed in LSTNet (Lai et al., 2018). For a fair comparison, we follow the same standard protocol (Liu et al., 2024) and split all forecasting datasets into training, validation, and test sets by the ratio of 6:2:2 for the ETT dataset and 7:1:2 for the other datasets. The characteristics of these datasets are shown in Table 1 (More can be found in the Appendix A.1).

Table 2. Multivariate forecasting results with prediction lengths $F \in \{96, 192, 336, 720\}$ and fixed look-back length $T = 96$. Results are averaged from all prediction lengths. Full results are listed in the Appendix.

Models	**Leddam Ours**		iTransformer (2024)		TimesNet (2023)		MICN (2023)		DLinear (2023)		PatchTST (2023)		Crossformer (2023)		TiDE (2023)		SCINet (2022)	
Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	**0.431**	**0.429**	0.454	0.447	0.458	0.450	0.561	0.535	0.456	0.452	0.449	0.442	0.624	0.575	0.434	0.437	0.486	0.467
ETTh2	**0.373**	**0.399**	0.383	0.407	0.414	0.427	0.587	0.525	0.559	0.515	0.387	0.407	0.942	0.684	0.611	0.550	0.954	0.723
ETTm1	**0.386**	**0.397**	0.407	0.410	0.400	0.406	0.392	0.414	0.403	0.407	0.387	0.400	0.513	0.509	0.403	0.427	0.411	0.418
ETTm2	**0.281**	**0.325**	0.288	0.332	0.291	0.333	0.328	0.382	0.350	0.401	0.283	0.327	1.219	0.827	0.293	0.336	0.310	0.347
Electricity	**0.169**	**0.263**	0.178	0.270	0.192	0.295	0.187	0.295	0.212	0.300	0.216	0.304	0.244	0.334	0.251	0.344	0.268	0.365
Solar-Energy	**0.230**	0.264	0.233	**0.262**	0.301	0.319	0.296	0.371	0.330	0.401	0.270	0.307	0.641	0.639	0.347	0.417	0.282	0.375
Traffic	0.467	0.294	**0.428**	**0.282**	0.620	0.336	0.542	0.315	0.625	0.383	0.555	0.362	0.550	0.304	0.760	0.473	0.804	0.509
Weather	**0.242**	**0.272**	0.258	0.279	0.259	0.287	0.243	0.299	0.265	0.317	0.259	0.281	0.259	0.315	0.271	0.320	0.292	0.363
1^{st} Count	7	6	1	2	0	0	0	0	0	0	0	0	0	0	0	0	0	0

Evaluation protocol. Following TimesNet (Wu et al., 2023), we use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as the core metrics for the evaluation. To fairly compare the forecasting performance, we follow the same evaluation protocol, where the length of the historical horizon is set as $T = 96$ for all models and the prediction lengths $F \in \{96, 192, 336, 720\}$. Detailed hyperparameters of Leddam can be found in the Appendix A.2.

Baseline setting. We carefully choose very recently EIGHT well-acknowledged forecasting models as our baselines, including 1) Transformer-based methods: iTransformer (Liu et al., 2024), Crossformer (Zhang & Yan, 2023), PatchTST (Nie et al., 2023); 2) Linear-based methods: DLinear (Zeng et al., 2023), TiDE (Das et al., 2023); and 3) TCN-based methods: SCINet (LIU et al., 2022), MICN (Wang et al., 2023), TimesNet (Wu et al., 2023).

4.2. Experiments Results

Quantitative comparison. Comprehensive forecasting results are listed in Table 2 with the best bold in red and the second underlined in blue. We leave full forecasting results in APPENDIX G to save place. The lower MSE/MAE indicates the better prediction result. It is unequivocally evident that Leddam has demonstrated superior predictive performance across all datasets except Traffic, in which iTransformer gets the best forecasting performance. Like PatchTST and iTransformer, Leddam employs a vanilla transformer encoder as its backbone, devoid of any structural modifications. It is noteworthy, however, that these three models consistently exhibit superior performance across all datasets. This at least partially indicates that, compared to intricately designed model architectures, superior representation of raw time series features, such as 'Whole Series Embedding' used in iTransformer and Leddam along with PatchTST's patches, may indeed constitute the pivotal factors for achieving more efficient time-series predictions. It is noteworthy that among these three models, PacthTST employs a channel-independent design, exclusively addressing intra-series variations without considering inter-series dependencies. iTransformer utilizes channel-wise self-attention to model inter-series dependencies but falls short of adequately capturing intra-series variations. Compared to both, the proposed Leddam incorporates 'Whole Series Embedding' and 'Auto-regressive Embedding' to model inter-series dependencies and intra-series variations. Consequently, Leddam demonstrates superior performance across various datasets. However, these three models still maintain a leading position over other models across most datasets. This aligns with our hypothesis that appropriately modeling inter-series dependencies and intra-series variations in multivariate time series is key for achieving more precise forecasting.

4.3. Model Analysis

Ablation study of dual attention module We conducted ablation experiments across five datasets, including ETTh1, Traffic, Electricity, Solar-Energy, and Weather, to validate the performance enhancement introduced by our 'Auto-regressive self-attention' and 'Channel-wise self-attention' components. 'Channel' means we only used 'Channel-wise self-attention'. 'Auto' means we only used 'Auto-regressive self-attention'. 'w/o All' means we simply replace the 'Auto-regressive self-attention' and 'Channel-wise self-attention' components with a linear layer.

We observe substantial performance improvements in Table 3 introduced by the 'Auto-regressive self-attention' and 'Channel-wise self-attention' components. 'Auto-regressive self-attention' brings an average of 19.02% of MSE to decrease across five datasets compared to using linear layer, and 'Channel-wise self-attention' achieves 21.09% improvement. Moreover, their synergistic integration yields further enhancements in model performance, getting an average

Table 3. Ablation of model structure across five datasets with prediction lengths $F = 96$, and input length $T = 96$.

Models	ETTm1		Electricity		Traffic		Weather		Solar-Energy	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
w/o All	0.350	0.368	0.197	0.275	0.644	0.389	0.195	0.235	0.306	0.330
w/o Auto	0.337	0.371	0.148	0.242	0.438	0.288	0.168	0.212	0.211	0.253
w/o Channel	0.335	0.369	0.152	0.242	0.467	0.285	0.167	0.212	0.226	0.263
Leddam (Ours)	**0.320**	**0.359**	**0.139**	**0.233**	**0.424**	**0.269**	**0.158**	**0.203**	**0.202**	**0.240**

Table 4. Predictive performance comparison of moving average kernel and trainable 1D convolutional kernel across four datasets. The prediction horizon is uniformly set at $F = 720$, while the input length $T = 96$.

Design	ETTh2		ETTm2		Electricity		Traffic		Solar-Energy	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
DLinear	0.831	0.657	0.554	0.522	0.245	0.333	0.645	0.394	0.356	0.413
LD_UTL	0.742	0.607	0.541	0.505	0.220	0.311	0.602	0.378	0.333	0.397
LD_TL	**0.684**	**0.576**	**0.521**	**0.488**	**0.209**	**0.302**	**0.584**	**0.345**	**0.313**	**0.376**

25.03% of MSE decrease, reaching an optimal level. This attests to the efficacy of both design elements and once again validates our hypothesis, namely, appropriately modeling inter-series dependencies and intra-series variations in multivariate time series can yield better predictive performance.

Superiority of Learnable Decomposition Module over Moving Average Kernel. To better emphasize the advantages of our Learnable Decomposition Module with weights initialized using a Gaussian distribution, in comparison to conventional Moving Average Kernel, we conducted an extensive experiment. Given that DLinear arguably represents the most prominent instantiation utilizing a Moving Average Kernel for trend information extraction, and achieves performance comparable to other state-of-the-art methods with the mere use of two simple linear layers, we select it as our baseline. For comparison, we substitute its Moving Average Kernel with our Learnable Decomposition Module, denoting the modified model as LD_TL if the kernel is set to trainable, else LD_UTL if the kernel is set to untrainable.

In comparison to a simple Moving Average Kernel, the Learnable Decomposition Module, as depicted in Table 4, consistently exhibits superior predictive performance across all four datasets regardless of its trainability. The untrainable version of the 1D convolutional kernel brings an average 7.28% of MSE decrease across five datasets compared to using a Moving Average Kernel, while the trainable version gives 11.98%. The obtained results conclusively demonstrate the superior efficacy of our Learnable Decomposition Module over a simple Moving Average Kernel, and the trainability of the kernel plays a significant role in its adaptability. We leave further analysis to APPENDIX B.1-B.4.

Decomposition result analysis. To further investigate why our Learnable Decomposition Module (LD) is a better time series decomposition solution than Moving Average Kernel (MOV), we conducted the following analysis on the seasonal part and trend part obtained.

Since the seasonal part represents repetitive patterns in the raw sequence, a good seasonal part should capture all major frequencies in the raw sequence. We separately calculated the amplitude similarity of the dominant frequencies (top 25%) between the seasonal part obtained by each method and the raw sequence, denoted as **FFT**. A better decomposition strategy should yield a seasonal part with higher similarity to the dominant frequencies of the raw sequence.

A good trend part should effectively capture the trend changes in the original sequence. Thus, we employed Dynamic Time Warping (DTW) (Mü, 2007) to compute the similarity between the raw sequence and the trend parts obtained from the two decompositions, denoted as **DTW**. A superior decomposition strategy should result in a trend part with higher DTW similarity to the raw sequence.

We selected **the final variate of each dataset**. The variates used are as follows.

- **ETT Dataset**: Oil Temperature (OT) every hour or 15 minutes.
- **Electricity**: hourly electricity consumption of the 321st (last) user.
- **Solar-Energy**: solar power production every 10 minutes of the 137th (last) PV plant.
- **Traffic**: hourly road occupancy rates measured by the 862nd (last) sensor.
- **Weather**: CO2 (ppm) collected every 10 minutes.

LD is pre-trained on task: input-96-forecast-720. To avoid cherry-picking, both metrics are computed by averaging the results calculated over the entire test dataset.

Table 5. Decomposition result analysis. Comparison of DTW and FFT for LD and MOV across eight datasets.

Kernel	LD (Ours)		MOV	
Dataset/Metric	DTW	FFT	DTW	FFT
Electricity	**0.643**	**0.942**	0.618	0.931
Solar_Energy	**0.910**	**0.781**	0.873	0.734
Traffic	**0.603**	**0.993**	0.563	0.991
Weather	**0.858**	**0.760**	0.846	0.691
ETTh1	**0.741**	**0.892**	0.724	0.852
ETTh2	**0.675**	**0.900**	0.652	0.887
ETTm1	**0.821**	**0.754**	0.808	0.717
ETTm2	**0.925**	**0.894**	0.908	0.759

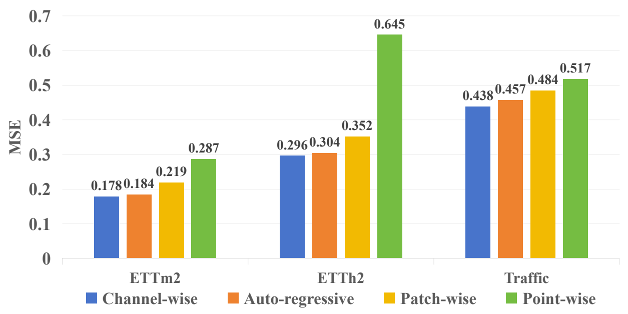


Figure 4. Predictive performance comparison(MSE) of 'Channel-wise self-attention', 'Auto-regressive self-attention', 'Patch-wise self-attention', and 'Point-wise self-attention' across ETTh2, ETTm2 and Traffic datasets. The prediction horizon is uniformly set at $F = 96$, while the input length $T = 96$.

The DTW and FFT of LD are consistently superior to that of the MOV across all eight datasets in Table 5. This demonstrates that LD is a better time series decomposition method compared to MOV.

Analysis of different attention mechanisms in inter-series dependencies modeling. We devised a comprehensive experiment to investigate three prevalent approaches for modeling inter-series dependencies: 'Channel-wise self-attention', 'Point-wise self-attention', and 'Patch-wise self-attention'. The first considers the entire sequence of a variate as a token (Liu et al., 2024), the second regards distinct variables at the same timestamp as tokens (Zhou et al., 2022a;c), and the third treats patches of the raw sequence as tokens (Zhang & Yan, 2023; Nie et al., 2023). Specifically, we eliminate the 'Auto-regressive self-attention' branch from the original Leddam structure to concentrate on inter-series dependencies modeling. We sequentially employ 'Channel-wise self-attention', 'Point-wise self-attention', and 'Patch-wise self-attention'. In Figure 4, it is readily apparent that compared to 'Point-wise self-attention', and 'Patch-wise self-attention', 'Channel-wise self-attention' exhibits superior predictive performance, implies a much better inter-series dependencies modeling ability.

Analysis of different attention mechanisms in intra-series variations modeling. Similarly, we replace the 'Channel-wise self-attention' branch from the original Leddam structure and use 'Point-wise self-attention', 'Patch-wise self-attention', and 'Auto-regressive self-attention' for intra-series variations modeling. The superiority of the 'Auto-regressive self-attention' architecture is proved by the experimental results in Figure 4.

4.4. Learnable Decomposition Generalization Analysis

In this subsection, we investigate the generalizability of the learnable decomposition module of **Leddam** by plugging it into other different kinds of models.

Model selection and experimental setting. To achieve this objective, we conduct experiments across a spectrum of representative time series forecasting model structures, including (1) Transformer-based methods: Informer (Zhou et al., 2022a), Transformer (Vaswani et al., 2017); (2) Linear-based methods: LightTS (Zhang et al., 2022); and (3) TCN-based methods: SCINet (LIU et al., 2022); (4) RNN-based methods: LSTM (Hochreiter & Schmidhuber, 1997). We standardize the input length T to 96, and similarly, the prediction length F is uniformly set to 96. Subsequently, comparative experiments were conducted on five datasets: ETTh2, ETTm2, Weather, Electricity, and Traffic. Specifically, we sequentially replaced the 'Auto-regressive self-attention' and 'Channel-wise self-attention' components in **Leddam** with each model. The comparative analysis was performed to assess the predictive performance before and after the incorporation of **Leddam**, comparing the original models with the augmented counterparts. And more results can be found in APPENDIX D.

Quantitative results. In Table 6, it is apparent that the incorporation of the **Leddam** structure leads to a notably substantial enhancement in the predictive performance of various models, even with the introduction of only a single linear layer. Specifically, LightTS demonstrates an average MSE reduction of 11.87% across five datasets, other models are LSTM: 48.56%, SCINet: 23.15%, Informer: 31.72%, and Transformer: 26.27%. Particularly noteworthy is the performance enhancement observed in the classical LSTM model, where the MSE experiences a remarkable decrease of 76.97% and 80.28% on ETTh2 and ETTm2, respectively, a profoundly surprising result. This unequivocally substantiates the generality of the **Leddam** structure.

Table 6. Improvements of LD over different models with prediction lengths $F = 96$, and fixed lookback length $T = 96$. LD means Learnable decomposition.

	Models	LightTS		LSTM		SCINet		Informer		Transformer	
	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	Original	0.439	0.464	1.537	0.987	0.784	0.680	2.234	1.239	1.528	1.029
	+ LD	**0.370**	**0.413**	**0.354**	**0.395**	**0.446**	**0.469**	**1.086**	**0.804**	**0.645**	**0.635**
	Improvement	**15.78%**	**10.97%**	**76.97%**	**59.99%**	**43.12%**	**31.04%**	**51.40%**	**35.10%**	**57.77%**	**38.30%**
ETTm2	Original	0.249	0.345	1.009	0.797	0.302	0.406	0.368	0.475	0.295	0.418
	+ LD	**0.201**	**0.293**	**0.199**	**0.299**	**0.220**	**0.322**	**0.302**	**0.402**	**0.287**	**0.399**
	Improvement	**19.05%**	**15.00%**	**80.28%**	**62.48%**	**27.03%**	**20.71%**	**17.91%**	**15.37%**	**2.58%**	**4.45%**
Weather	Original	0.166	0.234	0.221	0.302	0.214	0.290	0.201	0.285	0.167	0.248
	+ LD	**0.165**	**0.232**	**0.186**	**0.258**	**0.181**	**0.261**	**0.177**	**0.259**	**0.164**	**0.242**
	Improvement	**0.60%**	**0.77%**	**15.84%**	**14.57%**	**15.33%**	**9.96%**	**11.72%**	**9.00%**	**1.32%**	**2.14%**
Electricity	Original	0.233	0.337	0.305	0.384	0.247	0.345	0.374	0.441	0.275	0.368
	+ LD	**0.194**	**0.297**	**0.159**	**0.263**	**0.201**	**0.306**	**0.220**	**0.321**	**0.166**	**0.273**
	Improvement	**16.74%**	**11.79%**	**47.80%**	**31.44%**	**18.62%**	**11.30%**	**41.16%**	**27.18%**	**39.64%**	**25.81%**
Traffic	Original	0.640	0.405	0.680	0.375	0.668	0.427	0.806	0.453	0.739	0.397
	+ LD	**0.594**	**0.378**	**0.531**	**0.338**	**0.590**	**0.373**	**0.640**	**0.415**	**0.517**	**0.312**
	Improvement	**7.16%**	**6.60%**	**21.91%**	**9.87%**	**11.64%**	**12.65%**	**20.56%**	**8.45%**	**30.04%**	**21.41%**

5. Conclusion

Given the non-linear and intricate trend characteristics inherent in real-world time series data, this paper formulates a learnable convolution kernel as an improvement over the simple moving average kernel for time series decomposition. The Gaussian initialization and adaptable properties enable it to better align with the nuances of real-time series data, resulting in a more contextually fitting decomposition. Additionally, we present the dual attention module, incorporating both channel-wise self-attention and autoregressive self-attention. This innovative design facilitates the simultaneous capture of inter-series dependencies and intra-series variations with precision. Experimentally, our method achieves state-of-the-art performance and demonstrates remarkable framework generality, as supported by compelling analyses. In the future, we aim to optimize the application of our learnable convolutional kernel in the context of series decomposition.

Acknowledgements

This work was partially supported by the Start-up Fund of The Hong Kong Polytechnic University (No. P0045999) and the Seed Fund of the Research Institute for Smart Ageing (No. P0050946). We thank anonymous reviewers for their comments and suggestions.

Impact Statement

This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none of which we feel must be specifically highlighted here.

References

Dynamic Time Warping, pp. 69–84. Springer Berlin Heidelberg, Berlin, Heidelberg, 2007. ISBN 978-3-540-74048-3. doi: 10.1007/978-3-540-74048-3_4. URL https://doi.org/10.1007/978-3-540-74048-3_4.

Anonymous. FITS: Modeling time series with $10k$ parameters. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=bWcnvZ3qMb.

Anonymous. ModernTCN: A modern pure convolution structure for general time series analysis. In *The Twelfth International Conference on Learning Representations*, 2024b. URL https://openreview.net/forum?id=vpJMJerXHU.

Anonymous. Timemixer: Decomposable multiscale mixing for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024c. URL https://openreview.net/forum?id=7oLshfEIC2.

Das, A., Kong, W., Leach, A., Mathur, S. K., Sen, R., and Yu, R. Long-term forecasting with TiDE: Time-series dense encoder. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=pCbC3aQB5W.

Dong, J., Wu, H., Zhang, H., Zhang, L., Wang, J., and Long, M. SimMTM: A simple pre-training framework for masked time-series modeling. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=ginTcBUnL8.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2021. URL https://openreview.net/forum?id=YicbFdNTTy.

Han, L., Ye, H.-J., and Zhan, D.-C. The capacity and robustness trade-off: Revisiting the channel independent strategy for multivariate time series forecasting. Apr 2023.

Hochreiter, S. and Schmidhuber, J. Long short-term memory. *Neural Computation*, pp. 1735–1780, Nov 1997. doi: 10.1162/neco.1997.9.8.1735. URL http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Kim, T., Kim, J., Tae, Y., Park, C., Choi, J.-H., and Choo, J. Reversible instance normalization for accurate time-series forecasting against distribution shift. In *International Conference on Learning Representations*, 2022. URL https://openreview.net/forum?id=cGDAkQo1C0p.

Kitaev, N., Kaiser, L., and Levskaya, A. Reformer: The efficient transformer. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=rkgNKkHtvB.

Lai, G., Chang, W.-C., Yang, Y., and Liu, H. Modeling long-and short-term temporal patterns with deep neural networks. In *The 41st international ACM SIGIR conference on research & development in information retrieval*, pp. 95–104, 2018.

Li, S., Jin, X., Xuan, Y., Zhou, X., Chen, W., Wang, Y.-X., and Yan, X. Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In Wallach, H., Larochelle, H., Beygelzimer, A., d'Alché-Buc, F., Fox, E., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc., 2019. URL https://proceedings.neurips.cc/paper_files/paper/2019/file/6775a0635c302542da2c32aa19d86be0-Paper.pdf.

Li, Z., Qi, S., Li, Y., and Xu, Z. Revisiting long-term time series forecasting: An investigation on linear mapping. *arXiv preprint arXiv:2305.10721*, 2023.

LIU, M., Zeng, A., Chen, M., Xu, Z., LAI, Q., Ma, L., and Xu, Q. SCINet: Time series modeling and forecasting with sample convolution and interaction. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022. URL https://openreview.net/forum?id=AyajSjTAzmg.

Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A. X., and Dustdar, S. Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In *International Conference on Learning Representations*, 2022a. URL https://openreview.net/forum?id=0EXmFzUn5I.

Liu, Y., Wu, H., Wang, J., and Long, M. Non-stationary transformers: Exploring the stationarity in time series forecasting. In Oh, A. H., Agarwal, A., Belgrave, D., and Cho, K. (eds.), *Advances in Neural Information Processing Systems*, 2022b. URL https://openreview.net/forum?id=ucNDIDRNjjv.

Liu, Y., Li, C., Wang, J., and Long, M. Koopa: Learning non-stationary time series dynamics with koopman predictors. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=A4zzxu82a7.

Liu, Y., Hu, T., Zhang, H., Wu, H., Wang, S., and Long, M. itransformer: Inverted transformers are effective for time series forecasting. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=JePfAI8fah.

Ni, Z., Yu, H., Liu, S., Li, J., and Lin, W. Basisformer: Attention-based time series forecasting with learnable and interpretable basis. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023. URL https://openreview.net/forum?id=xx3qRKvG0T.

Nie, Y., Nguyen, N. H., Sinthong, P., and Kalagnanam, J. A time series is worth 64 words: Long-term forecasting with transformers. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=Jbdc0vTOcol.

Rangapuram, S. S., Seeger, M. W., Gasthaus, J., Stella, L., Wang, Y., and Januschowski, T. Deep state space models for time series forecasting. In Bengio, S., Wallach, H., Larochelle, H., Grauman, K., Cesa-Bianchi, N., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 31. Curran Associates, Inc., 2018. URL https://proceedings.neurips.cc/paper_files/paper/2018/file/5cf68969fb67aa6082363a6d4e6468e2-Paper.pdf.

Shao, Z., Zhang, Z., Wang, F., Wei, W., and Xu, Y. Spatial-temporal identity: A simple yet effective baseline for multivariate time series forecasting. In *Proceedings of the 31st ACM International Conference on Information & Knowledge Management*, pp. 4454–4458, 2022.

Taylor, S. J. and Letham, B. Forecasting at scale. *The American Statistician*, 72(1):37–45, 2018. doi: 10.1080/00031305.2017.1380080. URL https://doi.org/10.1080/00031305.2017.1380080.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L. u., and Polosukhin, I. Attention is all you need. In Guyon, I., Luxburg, U. V., Bengio, S., Wallach, H., Fergus, R., Vishwanathan, S., and Garnett, R. (eds.), *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL https://proceedings.neurips.cc/paper_files/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., and Xiao, Y. MICN: Multi-scale local and global context modeling for long-term series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=zt53IDUR1U.

Wu, D., Hu, J. Y.-C., Li, W., Chen, B.-Y., and Liu, H. STanhop: Sparse tandem hopfield model for memory-enhanced time series prediction. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=6iwg437CZs.

Wu, H., Xu, J., Wang, J., and Long, M. Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In Ranzato, M., Beygelzimer, A., Dauphin, Y., Liang, P., and Vaughan, J. W. (eds.), *Advances in Neural Information Processing Systems*, volume 34, pp. 22419–22430. Curran Associates, Inc., 2021. URL https://proceedings.neurips.cc/paper_files/paper/2021/file/bcc0d400288793e8bdcd7c19a8ac0c2b-Paper.pdf.

Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., and Long, M. TimesNet: Temporal 2d-variation modeling for general time series analysis. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=ju_Uqw384Oq.

Yi, K., Zhang, Q., Fan, W., He, H., Hu, L., Wang, P., An, N., Cao, L., and Niu, Z. FourierGNN: Rethinking multivariate time series forecasting from a pure graph perspective. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=bGs1qWQ1Fx.

Yi, K., Zhang, Q., Fan, W., Wang, S., Wang, P., He, H., An, N., Lian, D., Cao, L., and Niu, Z. Frequency-domain MLPs are more effective learners in time series forecasting. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023b. URL https://openreview.net/forum?id=iif9mGCTfy.

Zeng, A., Chen, M., Zhang, L., and Xu, Q. Are transformers effective for time series forecasting? In *Proceedings of the AAAI conference on artificial intelligence*, volume 37, pp. 11121–11128, 2023. URL https://ojs.aaai.org/index.php/AAAI/article/view/26317/26089.

Zhang, T., Zhang, Y., Cao, W., Bian, J., Yi, X., Zheng, S., and Li, J. Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. *arXiv preprint arXiv:2207.01186*, 2022. URL https://arxiv.org/abs/2207.01186.

Zhang, Y. and Yan, J. Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In *The Eleventh International Conference on Learning Representations*, 2023. URL https://openreview.net/forum?id=vSVLM2j9eie.

Zhao, Z., Chen, W., Wu, X., Chen, P. C. Y., and Liu, J. Lstm network: a deep learning approach for short-term traffic forecast. *IET Intelligent Transport Systems*, pp. 68–75, Mar 2017. doi: 10.1049/iet-its.2016.0208. URL http://dx.doi.org/10.1049/iet-its.2016.0208.

Zhou, H., Zhang, S., Peng, J., Zhang, S., Li, J., Xiong, H., and Zhang, W. Informer: Beyond efficient transformer for long sequence time-series forecasting. *Proceedings of the AAAI Conference on Artificial Intelligence*, pp. 11106–11115, Sep 2022a. doi: 10.1609/aaai.v35i12.17325. URL http://dx.doi.org/10.1609/aaai.v35i12.17325.

Zhou, T., MA, Z., wang, x., Wen, Q., Sun, L., Yao, T., Yin, W., and Jin, R. Film: Frequency improved legendre memory model for long-term time series forecasting. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 12677–12690. Curran Associates, Inc., 2022b.

Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., and Jin, R. Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. *CoRR*, abs/2201.12740, 2022c. URL https://arxiv.org/abs/2201.12740.

A. Experimental Details

A.1. Dataset Statistics

We elaborate on the datasets employed in this study with the following details.

- **ETT Dataset** (Zhou et al., 2022a) comprises two sub-datasets: **ETTh** and **ETTm**, which were collected from electricity transformers. Data were recorded at 15-minute and 1-hour intervals for ETTm and ETTh, respectively, spanning from July 2016 to July 2018.
- **Solar-Energy** (Lai et al., 2018) records the solar power production of 137 PV plants in 2006, which are sampled every 10 minutes.
- **Electricity1 Dataset**[1] encompasses the electricity consumption data of 321 customers, recorded on an hourly basis, covering the period from 2012 to 2014.
- **Traffic Dataset**[2] consists of hourly data from the California Department of Transportation. It describes road occupancy rates measured by various sensors on San Francisco Bay area freeways.
- **Weather Dataset**[3] contains records of 21 meteorological indicators, updated every 10 minutes throughout the entire year of 2020.

We follow the same data processing and train-validation-test set split protocol used in TimesNet (Wu et al., 2023), where the train, validation, and test datasets are strictly divided according to chronological order to make sure there are no data leakage issues. We fix the length of the lookback series as $T = 96$ for all datasets, and the prediction length $F \in \{96, 192, 336, 720\}$.

A.2. Implementation Details and Model Parameters

We trained our Leddam model using the L2 loss function and employed the ADAM optimizer. We initialized the random seed as $rs = 2021$. We also configured the hyperparameter $k = 25$-kernel size of the decomposition kernel (the Moving Average Kernel (MOV) and Learnable Decomposition Module (LD)). During the training process, we incorporated an early stopping mechanism, which would halt training after six epochs if no significant reduction in loss was observed on the validation set. For evaluation purposes, we used two key performance metrics: the mean square error (MSE) and the mean absolute error (MAE). We carried a grid hyperparameter search, where dimension of layer $dim \in \{256, 512\}$, learning rate $lr\ in\{0.001, 0.0001, 0.0005\}$, dropout ratio $dr \in \{0.0, 0.2, 0.5\}$ and number of network layers $nl \in \{1, 2, 3\}$. Our implementation was carried out in PyTorch and executed on an NVIDIA V100 32GB GPU. All the compared baseline models that we reproduced are implemented based on the benchmark of TimesNet (Wu et al., 2023) Repository, which is fairly built on the configurations provided by each model's original paper or official code. It is worth noting that both the baselines used in this paper and our Leddam have fixed a long-standing bug. This bug was originally identified in Informer (Zhou et al., 2022a) **(AAAI 2021 Best Paper)** and subsequently addressed by FITS (Anonymous, 2024a). For specific details about the bug and its resolution, please refer to **GitHub Repository**[4].

B. Further Analysis of Different Decomposition Methodologies

B.1. Decomposition Result Visualization

We present a detailed comparison of results obtained by decomposing various datasets using the Moving Average Kernel (MOV) and Learnable Decomposition Module (LD).

We still selected **the final variate of each dataset**. LD are pretrained on task: input-96-forecast-720. Given a sampling frequency of 10, 15 minutes, or 1 hour for these datasets, we opt to decompose time series of two lengths: 120 and 720, to better contrast the decomposition results and reflect the extracted seasonal and trend patterns respectively.

The decomposition performance of LD is consistently superior to that of the MOV across all eight datasets in Figure 5–12.

[1]`https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014`
[2]`https://pems.dot.ca.gov/`
[3]`https://www.bgc-jena.mpg.de/wetter/`
[4]`https://github.com/VEWOXIC/FITS`

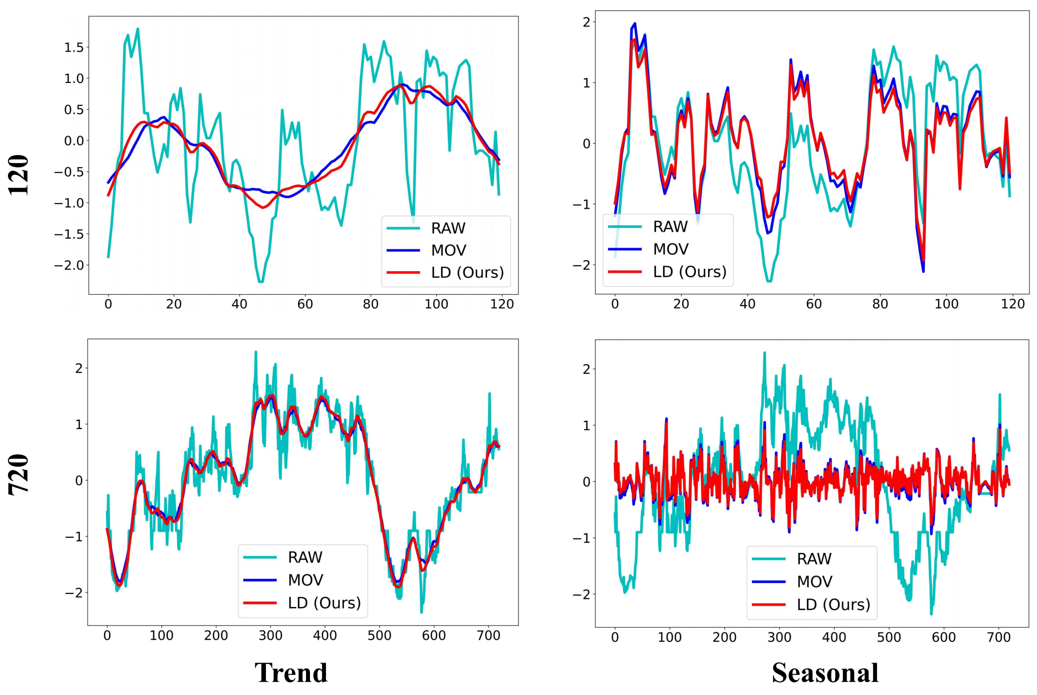


Figure 5. Trend-Seasonal Decomposition Results obtained by LD (Red) and MOV (Blue) on ETTh1.

B.2. Visualization of Weights

We present visualizations of the weight for the Learnable Decomposition Module (LD) and Moving Average Kernel (MOV) in Figure 13. It is observed that, unlike MOV, which employs uniform weights across all datasets, our LD adapts to the characteristics of different datasets, generating context-specific weights for sequence decomposition.

B.3. Impact of RevIN

To explore how much ReVIN (Kim et al., 2022) improves the performance of LD (and MOV), we utilized Autoformer (Wu et al., 2021) as the baseline and investigated the performance of LD and MOV under scenarios with and without ReVIN. In Table 7, We computed that ReVIN yielded an average MSE performance improvement (reduction) of 5.24% across all tasks for LD, and 9.04% for MAE. For MOV: 1.75% for MSE and 3.32% for MAE. Compared to MOV, ReVIN would bring greater performance improvements to LD.

B.4. Further Improvement of Multi-scale Hybrid Decomposition

In MICN (Wang et al., 2023), it is observed that they employ a multi-scale decomposition strategy akin to FEDformer (Zhou et al., 2022c). Specifically, they utilize decomposition kernels of varying scales, followed by integrating all decomposed outputs to derive the trend part:

$$\begin{aligned} i &= 1, \ldots, n \\ X^i_{Trend} &= AvgPool(Padding(X_{embed}))_{kernel_i} \\ X_{Trend} &= Intergrating(X^1_{Trend}, \ldots, X^n_{Trend}) \\ X_{Seasonal} &= X_{embed} - X_{Trend} \end{aligned} \tag{8}$$

MICN argues that using simple mean operations to integrate these different patterns is a superior plan.

We also conducted exploratory experiments to investigate the performance of our Learnable Decomposition Module (LD) in this particular context. We replaced the fundamental decomposition kernel employed in MICN, i.e., the basic Moving

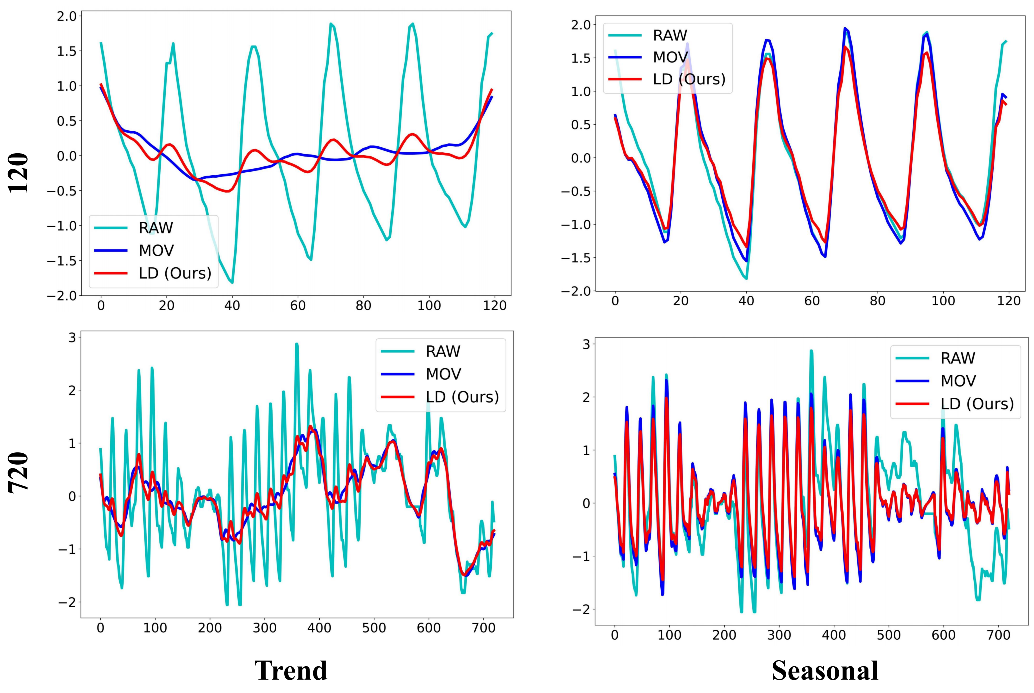


Figure 6. Trend-Seasonal Decomposition Results obtained by LD (Red) and MOV (Blue) on ETTh2.

Average Kernel (MOV), with our LD. Subsequently, comparative experiments were conducted on four datasets(ETTm2, Weather, Electricity, Traffic) with input length $T = 96$ and prediction length $F = 96$. The LD consistently outperforms MOV under a multi-scale decomposition strategy across all datasets in Table 8, which highlights the excellence of LD.

C. Further Analysis of Auto-regressive Self-attention

Our experiments in Table 9 demonstrate that our Auto-regressive Embedding, compared to Point-wise and Patch-wise Embedding, maintains crucial temporal positional information within permutation-invariant self-attention mechanisms. This is demonstrated by the low dependency of Auto-regressive Embedding on positional encoding.

D. Generalization Analysis of Leddam

To better illustrate the generality of our Leddam framework and its performance improvement across various models, we opted to visualize the predictive outcomes on three representative datasets (Electricity, ETTh2, Traffic) in Figure 14-18. For a prediction task with input length $T = 96$ and prediction length $F = 96$, we present a comparative analysis of the predictive performance of different models before and after integrating our Leddam framework. To achieve this objective, we conduct experiments across a spectrum of representative time series forecasting model structures, including (1) Transformer-based methods: Informer (Zhou et al., 2022a), Transformer (Vaswani et al., 2017); (2) Linear-based methods: LightTS (Zhang et al., 2022); and (3) TCN-based methods: SCINet (LIU et al., 2022); (4) RNN-based methods: LSTM (Hochreiter & Schmidhuber, 1997). After the introduction of the Leddam framework, various models have consistently demonstrated superior predictive performance.

E. Model Efficiency Study

We evaluated the **parameter count**, and the **inference time** (average of 5 runs on a single NVIDIA V100 32GB GPU) with $batch\_size = 1$ on **ETTh1** and **Electricity** dataset. We set the dimension of layer $dim \in \{96, 192, 336, 720\}$, and the number of network layers $nl = 2$. The task is input-96-forecast-720. We explored **Leddam** and four cutting-edge

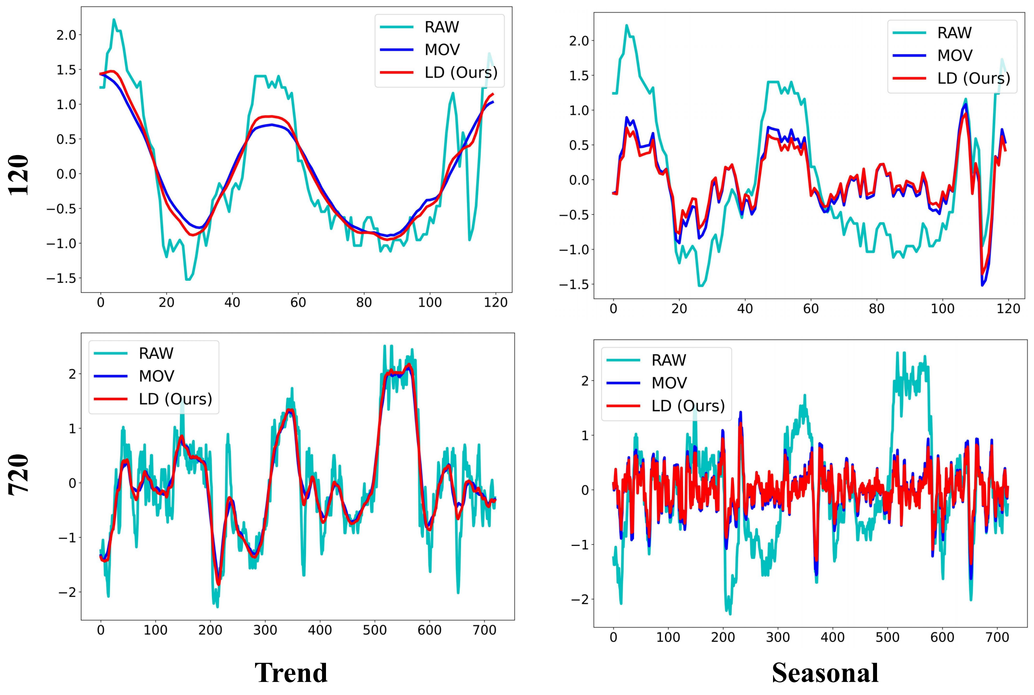


Figure 7. Trend-Seasonal Decomposition Results obtained by LD (Red) and MOV (Blue) on ETTm1.

transformer-based multivariate time series forecasting models:**iTransformer**, **Crossformer**, **PatchTST**, and **FEDformer**. Results can be found in Table 10.

F. Hyperparameter Sensitivity Analysis

We investigated the impact of the most significant parameters of Leddam: dimension of layer (dim), number of network layers (nl), dropout ratio (dr), and size of decomposition kernel (k). The default settings were: $dr = 0.0$, $dim = 512$, $nl = 2$, $k = 25$. Based on Table 11– 14, we can easily conclude that **Leddam** is insensitive to dropout ratio and kernel_size, while more sensitive to the number of network layers and the dimension of layers. For the kernel size, we argue after our initialization with a Gaussian distribution of 0 mean and 1 variance, the weight of the central element is the largest. As moving away from the center, the weights gradually decrease. Therefore, when the kernel size increases, the weights of the farther points become smaller, thus not significantly affecting the final decomposition result.

G. Full Forecasting Results

The full multivariate forecasting results are provided in the following section due to the space limitation of the main text. Table 15 contains the detailed results of eight baselines and our Leddam on eight well-acknowledged forecasting benchmarks.

H. Influence of Input Length on Prediction Performance

In principle, extending the look-back window increases the receptive field, leading to a potential improvement in forecasting performance. A robust Time Series Forecasting (TSF) model equipped with a strong temporal relation extraction capability should yield improved results with larger look-back window sizes (Zeng et al., 2023). As demonstrated in Figure 19, Our Leddam model consistently and effectively diminishes MSE scores as the receptive field expands, affirming its capacity to leverage longer look-back windows and superior temporal relation extraction capabilities.

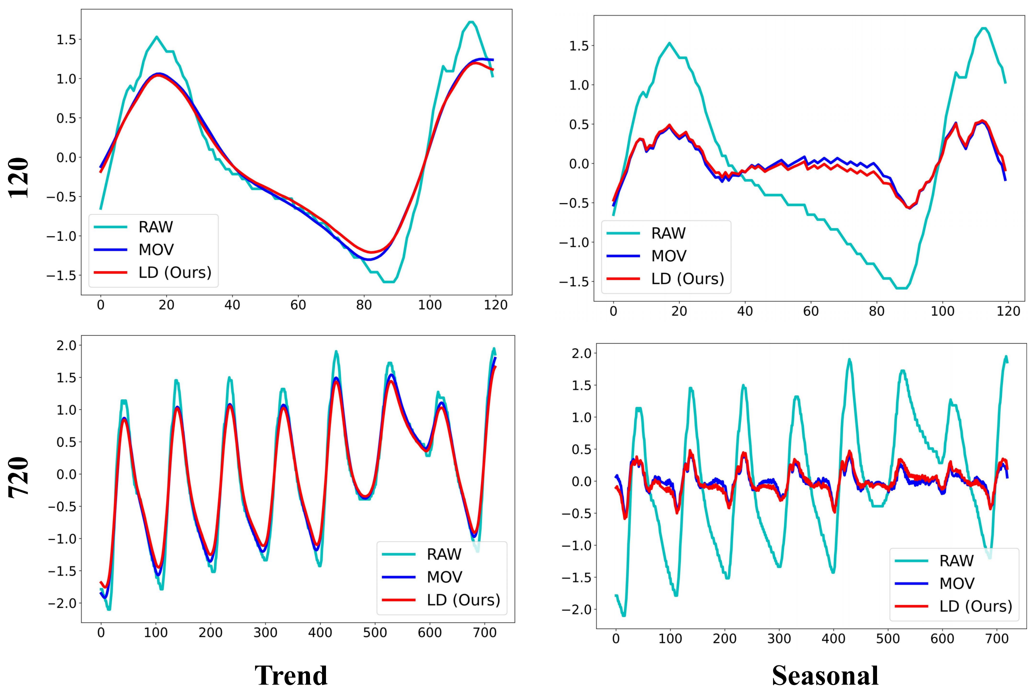


Figure 8. Trend-Seasonal Decomposition Results obtained by LD (Red) and MOV (Blue) on ETTm2.

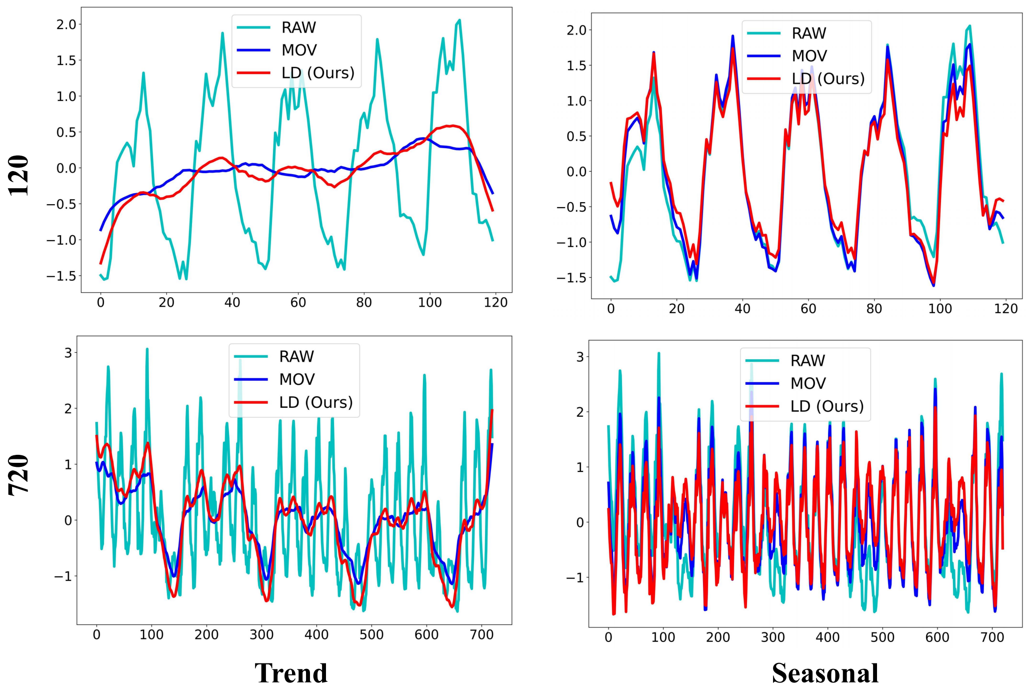


Figure 9. Trend-Seasonal Decomposition Results obtained by LD (Red) and MOV (Blue) on Electricity.

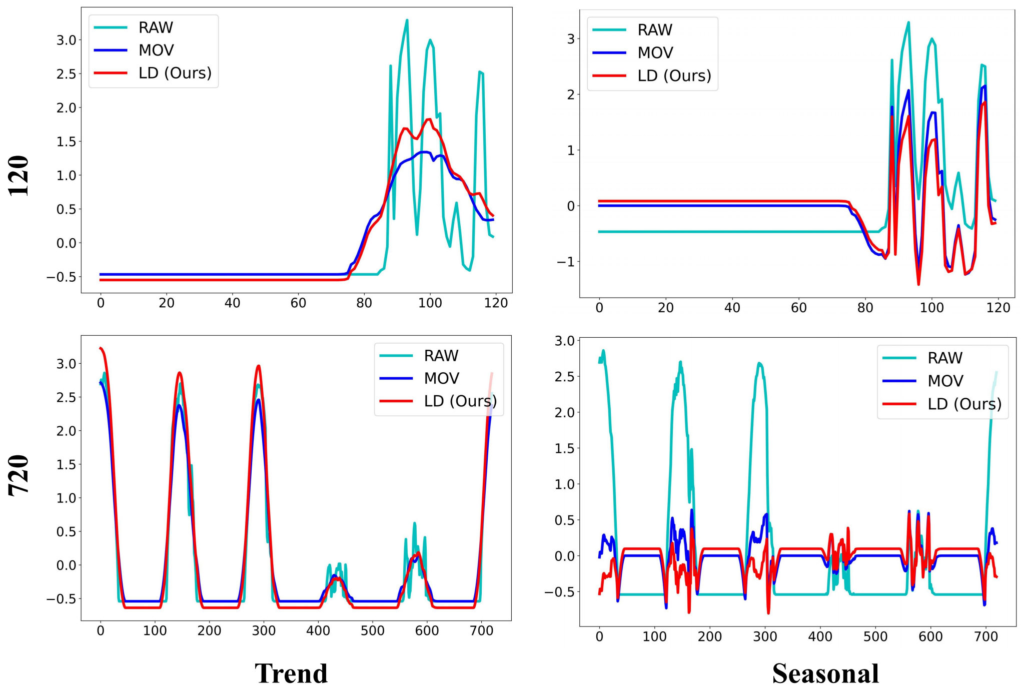


Figure 10. Trend-Seasonal Decomposition Results obtained by LD (Red) and MOV (Blue) on Solar-Energy.

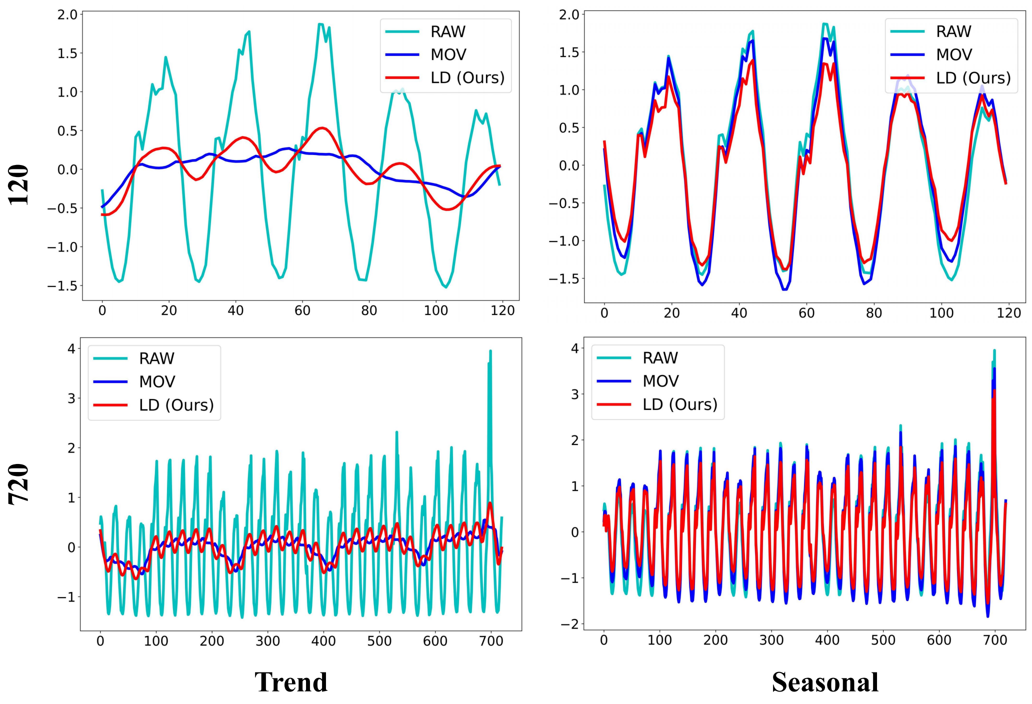


Figure 11. Trend-Seasonal Decomposition Results obtained by LD (Red) and MOV (Blue) on Traffic.

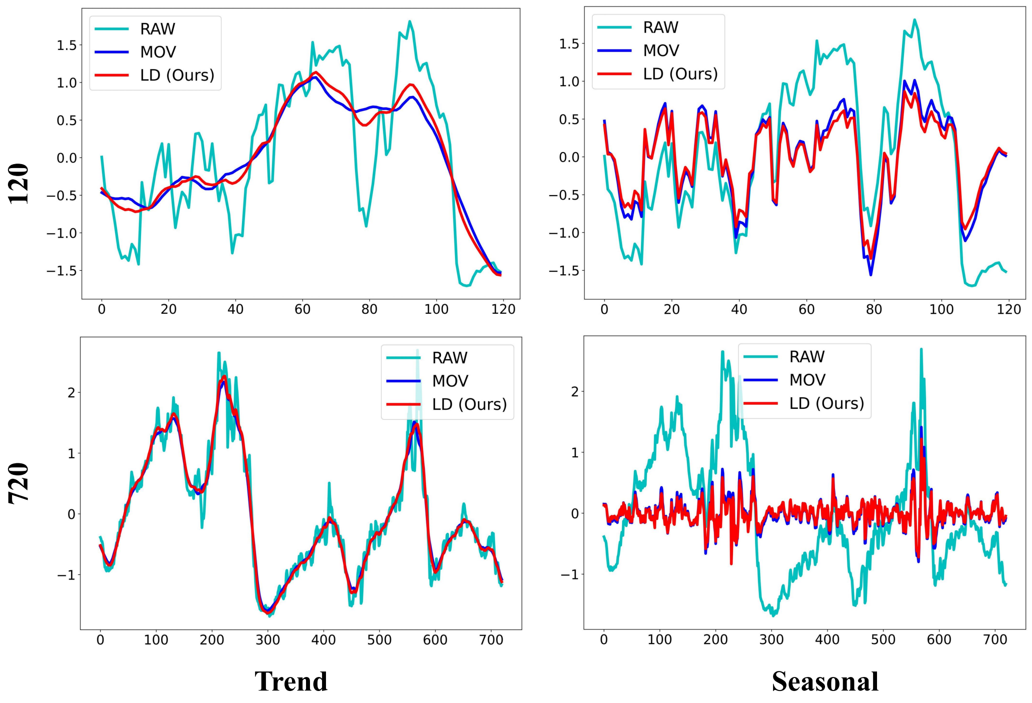


Figure 12. Trend-Seasonal Decomposition Results obtained by LD (Red) and MOV (Blue) on Weather.

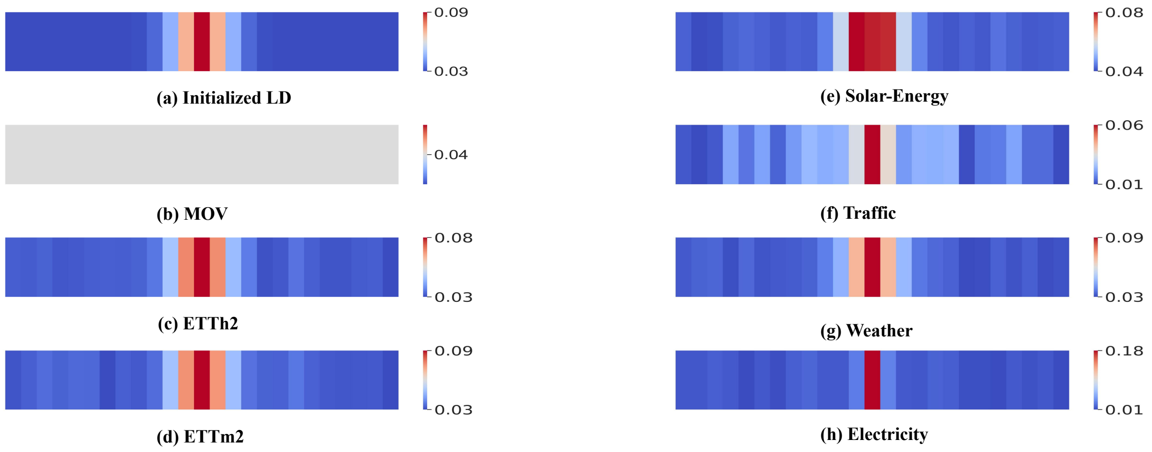


Figure 13. Weight Visualization of the Learnable Decomposition Module (LD) and Moving Average Kernel (MOV). (a) The weight of initialized LD and (b) MOV are the same for all datasets. (c)-(h) is the weight visualization of the LD after training on other datasets.

Table 7. Model Comparison

Dataset/Metric	pred_len	LD+ReVIN		MOV+ReVIN		LD		MOV	
		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTm1	96	0.667	0.526	0.694	0.538	0.664	0.531	0.671	0.534
	720	0.659	0.538	0.753	0.559	0.647	0.547	0.688	0.556
ETTm2	96	0.207	0.281	0.218	0.292	0.216	0.299	0.223	0.306
	720	0.414	0.404	0.422	0.41	0.435	0.43	0.426	0.423
Weather	96	0.201	0.249	0.217	0.266	0.289	0.366	0.233	0.304
	720	0.373	0.369	0.378	0.373	0.402	0.408	0.394	0.396

Table 8. Comparision of Learnable Decomposition Module (LD) and Moving Average Kernel (MOV) under multi-scale decomposition strategy across ETTm2, Weather, Electricity, and Traffic datasets with prediction lengths $F = 96$, and input length $T = 96$.

Design	ETTm2		Electricity		Traffic		Weather	
	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
Original	0.197	0.296	0.202	0.268	0.172	0.279	0.530	0.321
LD	**0.183**	**0.281**	**0.170**	**0.230**	**0.164**	**0.276**	**0.515**	**0.310**
Decrease	**6.87%**	**5.07%**	**15.97%**	**14.02%**	**4.71%**	**1.11%**	**2.83%**	**3.28%**

Table 9. Forecasting performance of Auto-regressive, Point-wise, and Patch-wise Embedding on the ETTh2, ETTm2, and Weather datasets. The forecasting task is input-96-forecast-96/720. **w/o** means without position embedding, **pos** means position embedding is used.

Models	pred_len	Type	Auto		Patch		Point	
Dataset/Metric			MSE	MAE	MSE	MAE	MSE	MAE
ETTh2	96	w/o	0.294	0.343	0.382	0.429	1.453	1.000
		pos	0.293	0.343	0.354	0.404	0.958	0.803
	720	w/o	0.413	0.432	0.926	0.703	3.533	1.625
		pos	0.416	0.433	0.835	0.666	1.569	0.991
ETTm2	96	w/o	0.176	0.257	0.226	0.321	0.419	0.511
		pos	0.175	0.256	0.208	0.314	0.335	0.447
	720	w/o	0.398	0.395	0.963	0.726	2.826	1.386
		pos	0.393	0.394	0.732	0.671	1.411	0.996
Weather	96	w/o	0.166	0.211	0.176	0.240	0.219	0.309
		pos	0.167	0.213	0.165	0.234	0.174	0.260
	720	w/o	0.342	0.343	0.355	0.405	0.351	0.395
		pos	0.342	0.344	0.337	0.370	0.346	0.388

Table 10. Model efficiency analysis. * means 'former.' **Para** means 'Parameter count(M).' **Time** means 'inference time(ms).'

Datasets/Models	dim	Leddam		PatchTST		Cross*		iTrans*		FED*	
		Param	Time	Para	Time	Para	Time	Para	Time	Para	Time
ETTh1	256	2.50	233.92	3.27	251.00	8.19	399.00	**1.27**	**177.67**	3.43	303.556
	512	9.20	249.34	8.64	266.66	32.11	445.74	**4.63**	**190.92**	13.68	345.736
Electricity	256	2.59	283.04	3.27	322.53	13.66	432.40	**1.27**	**192.12**	4.24	347.634
	512	9.36	296.70	8.64	411.96	43.04	507.54	**4.63**	**249.60**	15.29	398.599

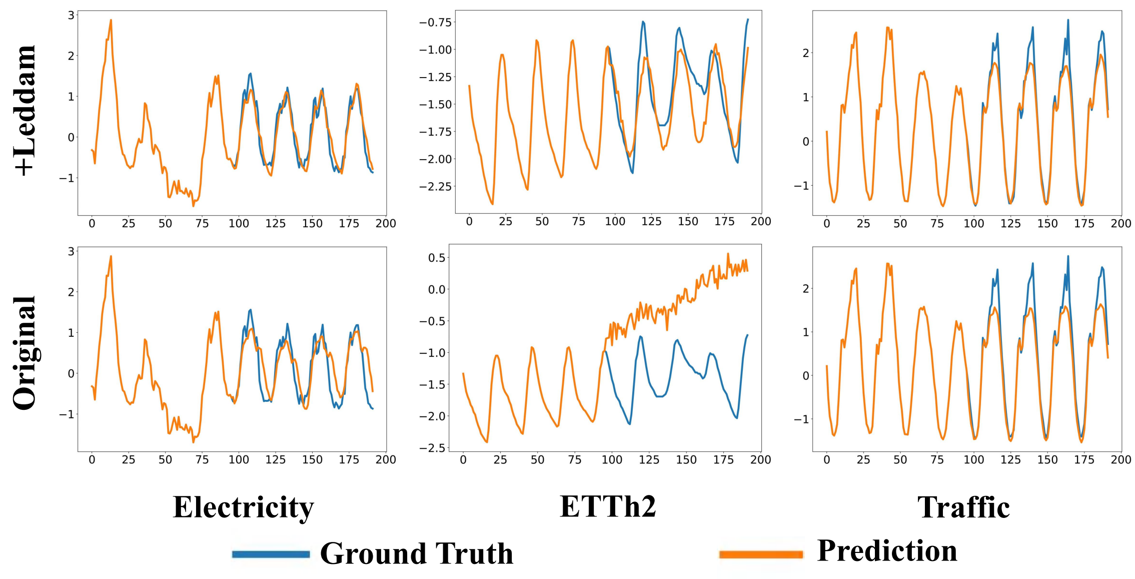


Figure 14. Visualization of input-96-predicts-96 results of LSTM (with and without Leddam) on three datasets (Electricity, ETTh2, Traffic). The above rows represent the performance incorporating our Leddam framework, while the below rows depict the predictive performance of the original model.

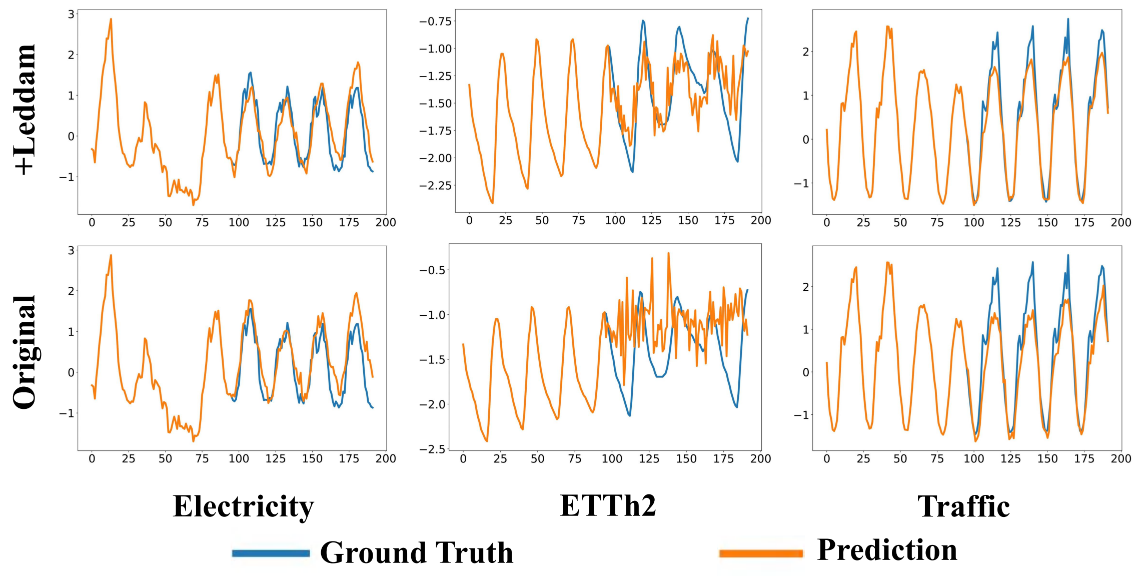


Figure 15. Visualization of input-96-predicts-96 results of SCINet (with and without Leddam) on three datasets (Electricity, ETTh2, Traffic). The above rows represent the performance incorporating our Leddam framework, while the rows below depict the predictive performance of the original model.

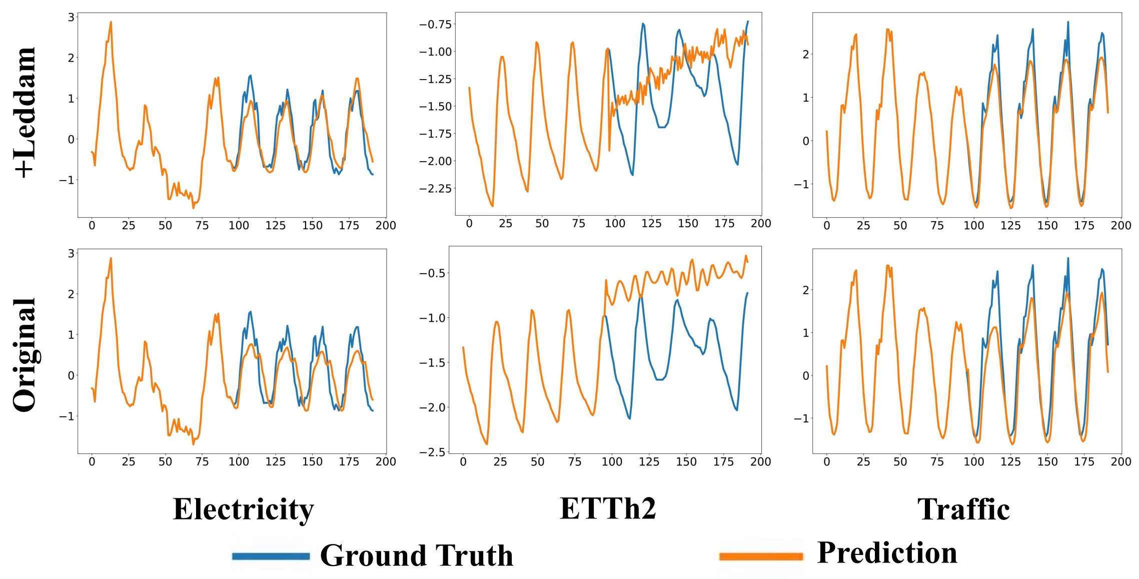


Figure 16. Visualization of input-96-predicts-96 results of Transformer (with and without Leddam) on three datasets (Electricity, ETTh2, Traffic). The above rows represent the performance incorporating our Leddam framework, while the rows below depict the predictive performance of the original model.

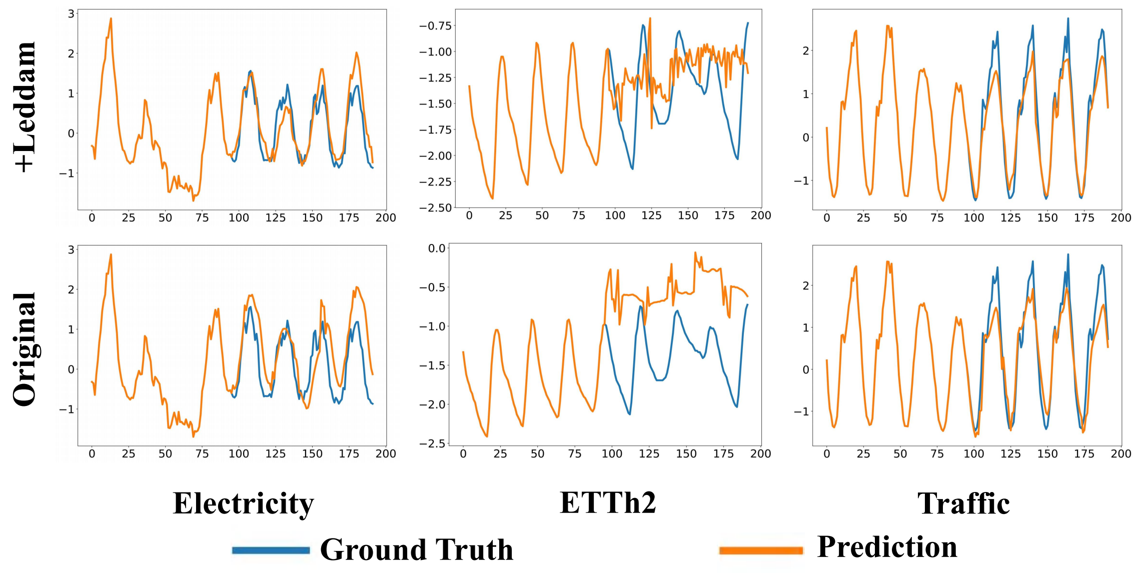


Figure 17. Visualization of input-96-predicts-96 results of Informer (with and without Leddam) on three datasets (Electricity, ETTh2, Traffic). The above rows represent the performance incorporating our Leddam framework, while the rows below depict the predictive performance of the original model.

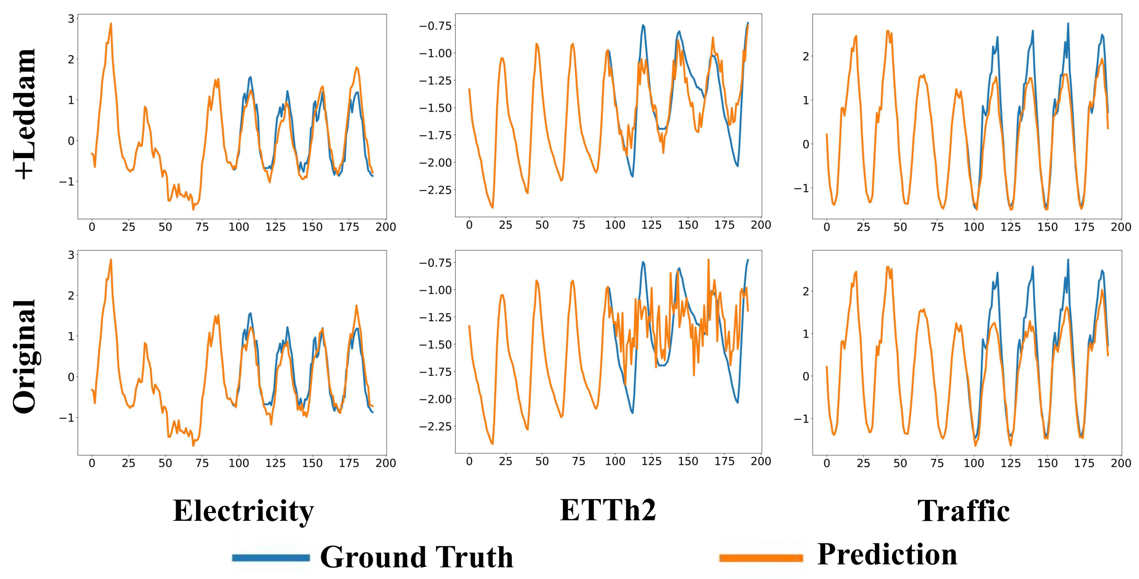


Figure 18. Visualization of input-96-predicts-96 results of LightTS (with and without Leddam) on three datasets (Electricity, ETTh2, Traffic). The above rows represent the performance incorporating our Leddam framework, while the rows below depict the predictive performance of the original model.

Table 11. Impact of dropout ratio (dr). $dr \in \{0.0, 0.1, 0.2, 0.5\}$

Dataset	ETTm1		ETTm2	
dr/Metric	MSE	MAE	MSE	MAE
0.0	0.469	0.447	0.406	0.400
0.1	0.468	0.446	0.404	0.401
0.2	0.476	0.452	0.406	0.401
0.5	0.474	0.452	0.409	0.404

Table 12. Impact of dimension of layer (dim). $dim \in \{128, 256, 512, 1024\}$

Dataset	ETTm1		ETTm2	
dim/Metric	MSE	MAE	MSE	MAE
128	0.486	0.454	0.399	0.397
256	0.473	0.449	0.403	0.399
512	0.469	0.447	0.406	0.400
1024	0.492	0.459	0.402	0.397

Table 13. Impact of number of network layers (nl). $nl \in \{1, 2, 3, 4\}$

Dataset	ETTm1		ETTm2	
nl/Metric	MSE	MAE	MSE	MAE
1	0.474	0.447	0.398	0.398
2	0.469	0.447	0.406	0.400
3	0.482	0.456	0.427	0.413
4	0.485	0.456	0.407	0.401

Table 14. Impact of kernel_size (k). $k \in \{15, 25, 55, 75, 105\}$

Dataset	ETTm1		ETTm2	
k/Metric	MSE	MAE	MSE	MAE
15	0.469	0.447	0.406	0.400
25	0.469	0.447	0.406	0.400
55	0.469	0.448	0.407	0.401
75	0.470	0.446	0.406	0.401
105	0.469	0.447	0.406	0.400

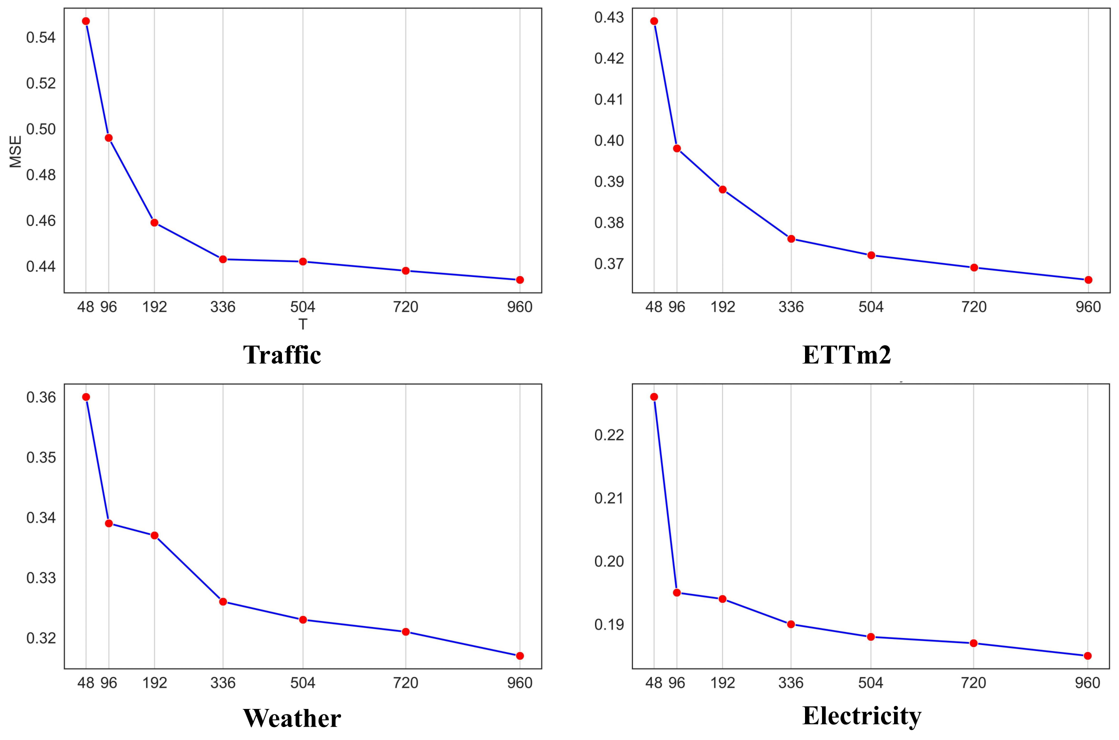


Figure 19. Forecasting performance (MSE and MAE) of Leddam with varying look-back windows on 4 datasets: ETTm2, Electricity, Traffic, and Weather. The look-back windows are selected to be $T \in \{48, 96, 192, 336, 504, 720, 960\}$, and the prediction horizons are $F = 720$.

Table 15. Multivariate long-term forecasting result comparison. We use prediction lengths $F \in \{96, 192, 336, 720\}$, and input length $T = 96$. The best results are in **bold** and the second bests are underlined.

Model		**Leddam (Ours)**		iTransformer (2024)		TimesNet (2023)		MICN (2023)		DLinear (2023)		PatchTST (2023)		Crossformer (2023)		TiDE (2023)		SCINet (2022)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.377**	**0.394**	0.386	0.405	0.384	0.402	0.421	0.431	0.386	0.400	0.378	0.396	0.420	0.439	0.377	0.397	0.404	0.415
	192	**0.424**	**0.422**	0.441	0.436	0.436	0.429	0.474	0.487	0.437	0.432	0.424	0.425	0.541	0.520	0.425	0.431	0.456	0.445
	336	**0.459**	**0.442**	0.487	0.458	0.491	0.469	0.569	0.551	0.481	0.459	0.466	0.448	0.722	0.648	0.461	0.443	0.519	0.481
	720	**0.463**	**0.459**	0.503	0.491	0.521	0.500	0.770	0.672	0.519	0.516	0.529	0.500	0.814	0.692	0.471	0.478	0.564	0.528
	Avg	**0.431**	**0.429**	0.454	0.447	0.458	0.450	0.561	0.535	0.456	0.452	0.449	0.442	0.624	0.575	0.434	0.437	0.486	0.467
ETTh2	96	**0.292**	**0.343**	0.297	0.349	0.340	0.374	0.299	0.364	0.333	0.387	0.302	0.348	0.745	0.584	0.400	0.440	0.707	0.621
	192	**0.367**	**0.389**	0.380	0.400	0.402	0.414	0.441	0.454	0.477	0.476	0.388	0.400	0.877	0.656	0.528	0.509	0.860	0.689
	336	**0.412**	**0.424**	0.428	0.432	0.452	0.452	0.654	0.567	0.594	0.541	0.426	0.433	1.043	0.731	0.643	0.571	1.000	0.744
	720	**0.419**	**0.438**	0.427	0.445	0.462	0.468	0.956	0.716	0.831	0.657	0.431	0.446	1.104	0.763	0.874	0.679	1.249	0.838
	Avg	**0.373**	**0.399**	0.383	0.407	0.414	0.427	0.587	0.525	0.559	0.515	0.387	0.407	0.942	0.684	0.611	0.550	0.954	0.723
ETTm1	96	**0.319**	**0.359**	0.334	0.368	0.338	0.375	0.324	0.375	0.345	0.372	0.322	0.362	0.360	0.401	0.347	0.384	0.350	0.385
	192	0.369	**0.383**	0.377	0.391	0.374	0.387	0.366	0.402	0.380	0.389	**0.366**	0.387	0.403	0.440	0.397	0.409	0.382	0.400
	336	**0.394**	**0.402**	0.426	0.420	0.410	0.411	0.408	0.426	0.413	0.413	0.396	0.404	0.543	0.528	0.417	0.430	0.419	0.425
	720	**0.460**	**0.442**	0.491	0.459	0.478	0.450	0.481	0.476	0.474	0.453	0.464	0.446	0.744	0.666	0.472	0.485	0.494	0.463
	Avg	**0.386**	**0.397**	0.407	0.410	0.400	0.406	0.392	0.414	0.403	0.407	0.387	0.400	0.513	0.509	0.403	0.427	0.411	0.418
ETTm2	96	**0.176**	**0.257**	0.180	0.264	0.187	0.267	0.179	0.275	0.193	0.292	0.178	0.260	0.259	0.349	0.192	0.274	0.201	0.280
	192	0.243	0.303	0.250	0.309	0.249	0.309	0.307	0.376	0.284	0.362	**0.242**	**0.301**	0.543	0.551	0.253	0.313	0.283	0.331
	336	**0.303**	**0.341**	0.311	0.348	0.321	0.351	0.325	0.388	0.369	0.427	0.304	0.344	1.038	0.715	0.315	0.352	0.318	0.352
	720	**0.400**	**0.398**	0.412	0.407	0.408	0.403	0.502	0.490	0.554	0.522	0.410	0.404	6.037	1.693	0.413	0.406	0.439	0.423
	Avg	**0.281**	**0.325**	0.288	0.332	0.291	0.333	0.328	0.382	0.350	0.401	0.283	0.327	1.219	0.827	0.293	0.336	0.310	0.347
Electricity	96	**0.141**	**0.235**	0.148	0.240	0.168	0.272	0.164	0.269	0.197	0.282	0.195	0.285	0.219	0.314	0.237	0.329	0.247	0.345
	192	**0.159**	**0.252**	0.162	0.253	0.184	0.289	0.177	0.285	0.196	0.285	0.199	0.289	0.231	0.322	0.236	0.330	0.257	0.355
	336	**0.173**	**0.268**	0.178	0.269	0.198	0.300	0.193	0.304	0.209	0.301	0.215	0.305	0.246	0.337	0.249	0.344	0.269	0.369
	720	**0.201**	**0.295**	0.225	0.317	0.220	0.320	0.212	0.321	0.245	0.333	0.256	0.337	0.280	0.363	0.284	0.373	0.299	0.390
	Avg	**0.169**	**0.263**	0.178	0.270	0.192	0.295	0.187	0.295	0.212	0.300	0.216	0.304	0.244	0.334	0.251	0.344	0.268	0.365
Solar_Energy	96	**0.197**	0.241	0.203	**0.237**	0.250	0.292	0.222	0.310	0.290	0.378	0.234	0.286	0.310	0.331	0.312	0.399	0.237	0.344
	192	**0.231**	0.264	0.233	**0.261**	0.296	0.318	0.277	0.343	0.320	0.398	0.267	0.310	0.734	0.725	0.339	0.416	0.280	0.380
	336	**0.241**	**0.268**	0.248	0.273	0.319	0.330	0.297	0.386	0.353	0.415	0.290	0.315	0.750	0.735	0.368	0.430	0.304	0.389
	720	0.250	0.281	**0.249**	**0.275**	0.338	0.337	0.390	0.445	0.356	0.413	0.289	0.317	0.769	0.765	0.370	0.425	0.308	0.388
	Avg	**0.230**	0.264	0.233	**0.262**	0.301	0.319	0.296	0.371	0.330	0.401	0.270	0.307	0.641	0.639	0.347	0.417	0.282	0.375
Traffic	96	0.426	0.276	**0.395**	**0.268**	0.593	0.321	0.519	0.309	0.650	0.396	0.544	0.359	0.522	0.290	0.805	0.493	0.788	0.499
	192	0.458	0.289	**0.417**	**0.276**	0.617	0.336	0.537	0.315	0.598	0.370	0.540	0.354	0.530	0.293	0.756	0.474	0.789	0.505
	336	0.486	0.297	**0.433**	**0.283**	0.629	0.336	0.534	0.313	0.605	0.373	0.551	0.358	0.558	0.305	0.762	0.477	0.797	0.508
	720	0.498	0.313	**0.467**	**0.302**	0.640	0.350	0.577	0.325	0.645	0.394	0.586	0.375	0.589	0.328	0.719	0.449	0.841	0.523
	Avg	0.467	0.294	**0.428**	**0.282**	0.620	0.336	0.542	0.315	0.625	0.383	0.555	0.362	0.550	0.304	0.760	0.473	0.804	0.509
Weather	96	**0.156**	**0.202**	0.174	0.214	0.172	0.220	0.161	0.229	0.196	0.255	0.177	0.218	0.158	0.230	0.202	0.261	0.221	0.306
	192	0.207	**0.250**	0.221	0.254	0.219	0.261	0.220	0.281	0.237	0.296	0.225	0.259	**0.206**	0.277	0.242	0.298	0.261	0.340
	336	**0.262**	**0.291**	0.278	0.296	0.280	0.306	0.278	0.331	0.283	0.335	0.278	0.297	0.272	0.335	0.287	0.335	0.309	0.378
	720	**0.343**	**0.343**	0.358	0.349	0.365	0.359	0.311	0.356	0.345	0.381	0.354	0.348	0.398	0.418	0.351	0.386	0.377	0.427
	Avg	**0.242**	**0.272**	0.258	0.279	0.259	0.287	0.243	0.299	0.265	0.317	0.259	0.281	0.259	0.315	0.271	0.320	0.292	0.363
1st Count		**31**	**30**	6	9	0	0	0	0	0	0	2	1	1	0	0	0	0	0