Computer Modeling in Engineering & Sciences

Tech Science Press

https://doi.org/10.32604/cmes.2026.082506

ARTICLE

SegTSF: Hierarchical Segment Learning For Lightweight Multivariate Time-Series ForeCasting

Hyunjun Park[1], Hee-Gook Jun[2], Seongyong Kim[3] and Dong-Hyuk Im[4,*]

[1]Department of Artificial Intelligence Application, Kwangwoon University, Seoul, Republic of Korea
[2]Finda, Seoul, Republic of Korea
[3]Division of Big Data AI, Hoseo University, Asan, Republic of Korea
[4]School of Information Convergence, Kwangwoon University, Seoul, Republic of Korea

*Corresponding Author: Dong-Hyuk Im. Email: dhim@kw.ac.kr

Received: 17 March 2026; Accepted: 09 June 2026; Published: 30 June 2026

ABSTRACT: Time-series forecasting can significantly aid decision-making in fields in which immediate action is required, such as power demand forecasting, financial market analysis, and traffic flow management. Transformer-based models achieve high forecasting accuracy by learning complex temporal patterns; however, their extensive parameters and substantial computational costs make practical deployment difficult in latency-sensitive environments. Therefore, lightweight models based on linear layers have recently been studied for improved efficiency. However, existing linear-based models have difficulty capturing local patterns and fail to reflect sudden volatility or fine-grained local trends, limiting their overall representational capacity. In this paper, SegTSF is proposed, a linear-layer-based lightweight model for multivariate time-series forecasting that improves forecasting performance by enhancing the representational capacity of linear layers while maintaining computational efficiency. First, SegTSF reconstructs the input time series into several subsequences in units of periods and explicitly models intra-period relationships with a linear layer to capture detailed temporal information. Second, it divides each subsequence into segment units and applies individual linear layers to the relationships within and between segments to capture local patterns and global trends. Third, each predicted subsequence is reconstructed into its original dimensions to complete the final forecast. Experimental results on benchmark datasets show that the proposed SegTSF achieves performance improvement compared with existing lightweight models while using fewer parameters in various environments. The findings of this study show that SegTSF achieves a balance between efficiency and forecasting performance through hierarchical segment-wise learning within a lightweight architecture.

KEYWORDS: Hierarchical learning; lightweight model; segment-based learning; multivariate time-series forecasting

1 Introduction

Time-series forecasting is one of the most fundamental tasks in data analysis. It is widely used in various practical applications such as power demand forecasting, financial market analysis, and traffic flow management, in which historical data serve as the basis for predicting future demands or key indices. Beyond their predictive capabilities, the true value of these forecasts lies in their ability to support critical decision-making processes. Therefore, a wide range of methodologies have been investigated to enhance forecasting accuracy, ranging from classical statistical approaches to recent sophisticated deep-learning models.

In recent years, deep-learning models based on transformer architectures have achieved significant breakthroughs in time-series forecasting. By introducing an attention mechanism, these models effectively

capture complex long-range dependencies within time-series data and enable parallel processing, thereby substantially improving their predictive accuracy. However, because of their structural complexity, these models typically involve millions of parameters, leading to high computational costs and memory consumption during the inference process. Consequently, these high-performance models require superior hardware specifications, posing clear limitations for deployment and operation in practical environments with strict efficiency requirements [1–3].

To overcome the constraints of complex models, such as Transformers, the current trend in time-series forecasting research is shifting toward lightweight models that drastically reduce parameter counts and computational costs while maintaining accuracy [4]. Models such as DLinear [5], FITS [6], and SparseTSF [7] have demonstrated the practical utility of lightweight architectures by significantly reducing the number of parameters and computational overhead using linear layers. However, time-series data inherently contain complex local patterns and localized trend variations that evolve over time. Simple linear transformations alone have inherent limitations in effectively modeling these relationships. Particularly, they fail to capture rapid volatility or position-dependent fine-grained patterns, ultimately leading to a deficiency in the model's overall representational capacity. Consequently, these structural constraints become a major cause of diminished predictive reliability in forecasting scenarios that require both accuracy and computational efficiency.

To address these linear-based model limitations, this paper proposes SegTSF, a lightweight model designed to maximize forecasting performance while maintaining a balance between representational capacity and computational efficiency. The core strategy of SegTSF is to use simple linear layers and organize them hierarchically to effectively decouple and learn the local and global trends in the data. To achieve this, the model first partitions time-series data into period-aligned subsequences defined as sets of observations sampled at regular periodic intervals and explicitly models intra-period structural dependencies using linear layers, a factor often overlooked by existing lightweight models, to capture nuanced temporal information. SegTSF performs hierarchical segment learning by subdividing each subsequence into segment units. In the first stage, intra-segment learning, localized patterns within short segments are independently learned to effectively capture rapid volatility or local patterns that global linear mapping tends to miss. In the subsequent stage, inter-segment learning, correlations between the learned local features are identified to construct a comprehensive global trend. This architecture was designed to significantly enhance the representational capacity of the linear layers while minimizing model complexity, thereby providing reliable forecasting with improved expressiveness within a lightweight architecture.

The main contributions of this study can be summarized as follows:

- SegTSF is proposed, a lightweight model specifically designed to enhance multivariate time-series forecasting performance through hierarchical segment-wise learning.
- SegTSF explicitly captures intra-period structural dependencies and hierarchically learns local and global trends through segment-wise trend modeling, thereby enhancing the representational capacity of linear layers without introducing nonlinear operations.
- Experimental results on five benchmark datasets (ETTh1, ETTh2, Electricity, Weather, and Traffic) demonstrate that SegTSF achieves competitive forecasting accuracy compared with both complex and lightweight baselines while requiring significantly fewer parameters in several cases, indicating that the proposed hierarchical architecture effectively balances representational capacity and computational efficiency.

While individual techniques such as period-based reshaping, segmentation, and linear mappings have been explored in prior works, SegTSF uniquely integrates these components into a hierarchical architecture

that simultaneously captures intra-period structural dependencies, local patterns within segments, and global trends across segments. This specific combination and its resulting capability represent the core architectural contribution of the proposed model.

The remainder of this paper is organized as follows. Section 2 reviews related work. Section 3 provides a detailed description of the SegTSF architecture, including intra-period learning and hierarchical segment learning. Section 4 presents the experimental setup, quantitative results, and an efficiency comparison. Finally, Section 5 concludes the paper and suggests directions for future research.

2 Related Work

In this section, prior research underpinning SegTSF is reviewed. Existing time-series forecasting methodologies are described, with some emphasis on those that have been researched for lightweight models.

2.1 Time-Series Forecasting Models

Many time-series forecasting methodologies aim to improve forecasting accuracy by accurately capturing trends and seasonality, which are the core components of a time series. However, for cases with long data lengths and numerous variables, methodologies with increasingly complex architectures have been studied to capture correlations. Time-series forecasting has been widely applied across diverse domains, including financial markets. Recent surveys have examined neural network-based approaches for stock price forecasting, highlighting the growing importance of deep learning methods in financial time-series analysis [8].

Classical Statistical Approaches. Classical statistical approaches, such as ARIMA [9], Exponential Smoothing, and Prophet, make predictions by designing mathematical functions based on past numerical patterns, assuming that time-series data follow specific statistical regularities. They decomposed and interpreted the series into trend, seasonal, and residual components. These methodologies are characterized by their ability to mathematically explain how forecasting results are derived, and have very low computational loads. However, most classical methodologies have limitations in terms of capturing complex and nonlinear patterns and struggle to handle multivariate data, which has motivated research on deep-learning-based methods [10].

Recurrent Neural Networks (RNNs). RNNs integrate past time-series information into hidden states to process them effectively, and have been widely used in time-series tasks. However, their inherently sequential structure increases the inference latency, and the discrepancy between teacher-forced training during computational complexity and autoregressive decoding at the test time can induce exposure bias and multistep error accumulation. Although Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU) gating alleviate long-dependency issues, the latency and forecasting stability remain practical challenges [11].

Convolutional Neural Networks (CNNs). CNNs are effective at capturing local patterns and extracting meaningful features. Applying one-dimensional (1D) convolutions enables parallel training and inference, thereby delivering excellent performance in short forecasting horizons. However, because there is no recurrent state, long-range dependencies depend heavily on the designed receptive field, and adaptation can be unstable when the periodicity is weak, shifting, or under complex trends mixed with abrupt changes [12]. In contrast, SegTSF deliberately avoids convolutional operations in favor of pure linear transformations, prioritizing parameter efficiency and inference speed while capturing local patterns through its hierarchical segment structure.

Transformers. As the complexity of multivariate time-series data and the demand for long-term prediction increase, a Transformer architecture that introduces an attention mechanism to overcome the limitations of existing recurrent structures has been proposed. This design enables parallel processing by removing sequential operations from the encoder-decoder framework, and has achieved innovative progress in long-term dependency modeling by directly calculating correlations between all time points in a sequence via self-attention [13]. Early Transformer-based studies such as Informer [14] and Autoformer [15] focused on reducing the computational complexity of the attention mechanism, followed by the development of models such as PatchTST [16], which processes data in patch units. These models achieve high accuracy at the cost of including millions of parameters, thus requiring substantial computational resources and memory footprint during the inference process. These resource-intensive characteristics make deployment difficult in practical environments with strict efficiency requirements, raising the need to develop lightweight architectures for practical time-series forecasting [1].

2.2 Lightweight Forecasting Models

As discussed in Section 2.1, each architecture exhibits a clear tradeoff between the ability to capture the core characteristics of time-series data and computational efficiency. In particular, the computational complexity and operational speed of a model are crucial indicators that determine its adaptability to environments requiring high computational efficiency.

Specifically, RNNs have a limitation in that they cause inference latency due to their sequential computational structure reaching $O\left(L \cdot d^2\right)$, where L denotes the look-back window and d represents the hidden dimension. This structure makes it difficult to use GPU acceleration for parallel processing. By contrast, while Transformers enable parallel processing through the attention mechanism, their computational load and memory footprint increase exponentially in proportion to the square of the sequence length, $O\left(L^2 \cdot d\right)$. These cost issues are decisive factors hindering the practical deployment of deep-learning models in practical environments with strict efficiency requirements.

To overcome these structural constraints, research trends have recently shifted toward lightweight forecasting models that drastically suppress complexity to the $O\left(L\right)$ level using a single linear layer while maintaining or even improving prediction accuracy. These models can be used as practical alternatives to resource-intensive, high-performance deep-learning architectures by adopting efficient computational methods that reflect the structural characteristics of the time series.

Linear models that outperform complex architectures. DLinear emerged by pointing out that the self-attention mechanism of Transformers can lose temporal-order information and achieve high accuracy while minimizing model complexity. This model explicitly decomposes the input time series into trend and seasonality components using a moving average technique. It then performs direct multistep forecasting by applying independent linear layers to each component [5]. This simple structure not only surpassed complex Transformer models in terms of accuracy on nine benchmark datasets but also significantly improved computational efficiency.

Frequency-domain modeling. Another strategy for improving efficiency by compressing data is modeling in the frequency domain. FITS converts time-series data into a complex frequency domain using a real Fast Fourier Transform (rFFT). It then learns the amplitude and phase of the signal via a complex linear layer, and forecasts future time points through frequency interpolation [6]. In particular, by using a low-pass filter to remove high-frequency components and compress the data representation, it demonstrated excellent performance with only 5–10 k parameters.

Periodic downsampling. An approach to simplify the model via data resolution adjustment has also been proposed. SparseTSF downsamples data periodically to reconstruct a time series into a two-dimensional structure and focuses on inter-period trend forecasting. This strategy allows competitive accuracy in long-term forecasting while suppressing the number of parameters to less than 1 k [7].

Other recent lightweight approaches include TiDE [17], which adopts a multi-layer perceptron architecture for long-term forecasting, RLinear [18], which revisits simple linear models with normalization strategies, and TSMixer [19], which applies MLP-mixing operations across time and feature dimensions. While these models demonstrate strong performance, they involve higher computational complexity compared to SegTSF, which maintains a minimal parameter budget through its hierarchical segment structure.

In summary, existing lightweight models have achieved high efficiency through linear layers using various techniques. However, the short-term forecasting performance depends on how well local patterns, including rapid slope changes and small phase shifts within the recent window, are preserved. However, the current linear models have several limitations.

First, various data-processing methods adopted for lightweighting (such as abstraction, decomposition, and downsampling) inevitably involve data information loss in the process of increasing computational efficiency. Most lightweight models focus on capturing the global trend or low-frequency components of a time series to minimize computational complexity. In this process, intra-period dependencies, such as fine-grained variations or local correlations within individual periods, are overlooked or lost. Consequently, this degrades the ability to reproduce the fine patterns that determine the forecasting accuracy.

Second, most linear models perform global mapping by applying a single weight matrix W across the entire input window. However, actual time series often exhibit rapid fluctuations in specific segments. Global linear layers tend to 'average out' these localized rapid changes, limiting their capacity to represent fine-grained trend variations at specific positions [5–7].

These limitations highlight a fundamental tradeoff in existing methods for preserving local patterns and maintaining computational efficiency. Therefore, preserving these critical patterns while adhering to strict budgets for parameter count, computation, memory, and inference latency presents a significant challenge for accurate and efficient forecasting. To overcome these limitations and achieve an optimal balance between efficiency and expressiveness, this paper proposes SegTSF, which uses a hierarchical segment structure [5,7,20].

3 Proposed Method

This section presents SegTSF, a model designed to improve forecasting performance through hierarchical segment learning. This section begins by presenting the motivation for the proposed approach through an analysis of existing methods. Next, the preparation technique, including intra-period relational learning and segment-level hierarchical learning, which form the basis of the proposed SegTSF model, is discussed.

3.1 Preliminaries

3.1.1 Terminology

The primary notation used in this paper is presented in Table 1. The input and output time series are denoted as X and Y, respectively. The input lookback window length is L, and the forecast horizon length is H. A known period within the series is represented by w and the number of repetitions of the period in the lookback window is represented by n. Segmentation is defined by three parameters: k: the length of an individual segment; s: a single segment; and S: the number of segments within a subsequence.

Table 1: Notation and terminology.

Notation	Terminology
X	Input time-series
Y	Output time-series
L	Lookback window
H	Forecast horizon
w	A known periodicity
n	Number of iterations of Period in the lookback window
k	A scalar value representing the size of each segment
s	Single segment
S	Number of segments

3.1.2 Direct Multistep Forecasting

Multistep forecasting models are primarily categorized as iterated or direct. The iterated approach predicts future values sequentially, one step at a time, which incurs a high computational cost and suffers from error accumulation. By contrast, the direct approach generates the entire forecast horizon in a single forward pass. This strategy facilitates parallel computation, accelerates inference, and alleviates cumulative errors; thus, it has been widely adopted in Transformer- and linear-based models. To ensure computational efficiency and minimize inference latency, SegTSF adopts a direct multistep formulation [21].

3.1.3 Channel Independence

Modeling the interdependencies among channels in a multivariate time series is computationally expensive. To maintain a lightweight architecture, SegTSF adopts a channel-independent strategy. In this paradigm, each channel of a multivariate time series is treated as a distinct univariate series and a single model with parameters shared across all channels is used for forecasting. This parameter-sharing strategy substantially reduces the parameter count, mitigating the risk of overfitting while enabling the model to focus on unique temporal patterns within each channel [22].

3.1.4 Segmentation

Segmentation is a technique that processes time-series data by partitioning them into subsections of fixed length. Instead of mapping the entire lookback window of length L into a single pass, the model divides the data into segments of length k. This structures the computational units based on these segments, thereby effectively reducing the computational complexity of the model to the L/k level [15]. Furthermore, by enabling the model to focus on individual subsections rather than being submerged in the average flow of the entire sequence, it provides a theoretical foundation for explicitly capturing fine-grained local information that global mapping tends to miss.

3.2 Architecture

Fig. 1 illustrates the overall SegTSF architecture. The core contribution of the model is the learning of intra-period relationships and segment-wise hierarchical trends from period-aligned subsequences, which are reorganized sequences of data points sharing the same phase within a period.

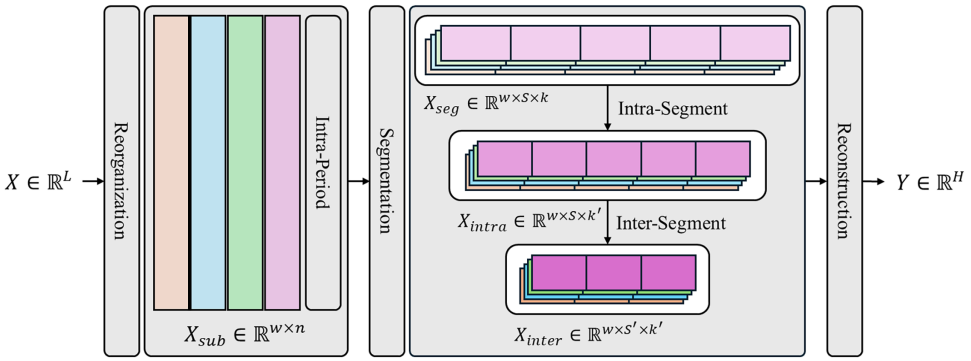

Figure 1: Overview of SegTSF architecture.

The SegTSF architecture consists of three sequential components. First, an intra-period linear layer models the structural dependencies among period-aligned subsequences. Second, a two-stage hierarchical segment module applies an intra-segment linear layer to capture local patterns within each segment, followed by an inter-segment linear layer to learn global trends across segments. Finally, a parameter-free reconstruction step reshapes the output back to the original time-series dimensions. All components are implemented without nonlinear activation functions between stages, ensuring minimal inference cost. The expressiveness of SegTSF stems not from nonlinearity, but from the structured decomposition of temporal patterns across period-aligned and segment-level representations.

3.2.1 Intra-Period

The input time series is reorganized into w subsequences to form subsequence matrix $X_{sub} \in \mathbb{R}^{w \times n}$, each containing values from the same position within a period. This process reconstructs the raw one-dimensional time series into a two-dimensional structure, effectively grouping data points that share the same periodic phase. By aligning points with identical offsets, the model can more easily capture the overarching global trend by decoupling it from intra-period seasonal fluctuations. Furthermore, this reorganization reduces the sequence length to be processed to n, thereby significantly enhancing the model's computational efficiency [7]. As in Eq. (1), a linear layer learns the relationships among these w subsequences within X_{sub}, thereby leveraging structural periodic information. This operation maps interactions across period indices while sharing weights across channels and bounding the parameter complexity to $O(w^2)$. Here, w denotes the period length, $n = L/w$ denotes the number of period repetitions in the lookback window, and $W_{period} \in \mathbb{R}^{w \times w}$ is the learnable weight matrix that captures intra-period structural dependencies.

$$X_{sub} \in \mathbb{R}^{w \times n} = \text{Reorganize}\,(X) \cdot W_{period},\ W_{period} \in \mathbb{R}^{w \times w} \tag{1}$$

3.2.2 Segment Partition

The core of SegTSF's hierarchical learning lies in partitioning each n-long subsequence X_{sub} into non-overlapping segments. Given hyperparameter k (segment length), each subsequence is divided into $S = n/k$ segments to form segment matrix $X_{seg} \in \mathbb{R}^{w \times S \times k}$. The model then performs a two-stage process: intra- and inter-segment learning.

Intra-Segment Learning. As shown in Eq. (2) and Fig. 2, the first linear layer independently learns the internal pattern of an input segment vector $s \in \mathbb{R}^k$ from X_{seg}, producing an encoded feature vector $s' \in \mathbb{R}^{k'}$. By processing all S segments through this layer, the model generates the intra-segment feature

matrix $X_{intra} \in \mathbb{R}^{w \times S \times k'}$. This enables the model to capture short local trend patterns such as brief decreases or increases embedded within the overall trend [23]. This layer is a projection along the segment-length axis, and the same weights are applied to each segment, thereby constraining the number of parameters to $O(k \cdot k')$. Nonlinearity was not used, which minimized the inference cost and latency. Here, k denotes the input segment length, k' denotes the output segment length, and $W_{intra} \in \mathbb{R}^{k \times k'}$ is a shared weight matrix applied identically across all S segments.

$$s' \in \mathbb{R}^{k'} = s \cdot W_{intra}, s \in \mathbb{R}^{k}, W_{intra} \in \mathbb{R}^{k \times k'} \tag{2}$$

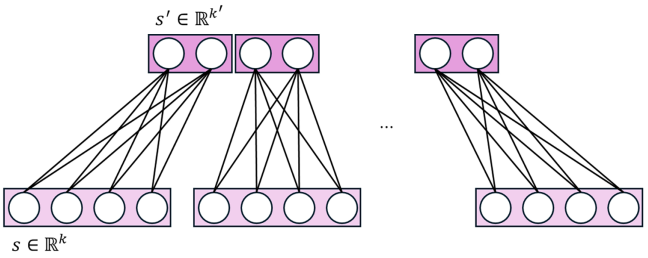

Figure 2: Intra-segment linear mapping.

Inter-Segment Learning. As shown in Fig. 3 and Eq. (3), the second linear layer learns the relationships among the S segments by processing feature matrix $X_{intra}^{(i)} \in \mathbb{R}^{S \times k'}$, generating output feature matrix $X_{inter}^{(i)} \in \mathbb{R}^{S' \times k'}$. Thus, the model learns the overall trend from these segments. The w predicted subsequences are then upsampled to restore the original time-series form, yielding the final forecast sequence Y. This layer is a linear map along the segment count axis comprising a global trend via inter-segment correlations. The number of parameters is $O(S \cdot S')$, and S' controls the global summarization resolution. Here, $S = n/k$ denotes the number of input segments, $S' = m/k'$ denotes the number of output segments with $m = H/w$, and $W_{inter} \in \mathbb{R}^{S \times S'}$ learns the global trend across segments.

$$X_{inter} \in \mathbb{R}^{S' \times k'} = X_{intra} \cdot W_{inter}, X_{intra} \in \mathbb{R}^{S \times k'}, W_{inter} \in \mathbb{R}^{S \times S'} \tag{3}$$

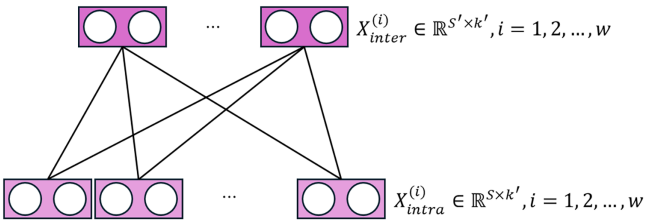

Figure 3: Inter-segment linear mapping.

The predicted subsequence length $m = H/w$ is determined by the product of the output dimensions of the two layers, i.e., $m = k' \times S'$. Therefore, in the model design, k' and S' are chosen from the common divisors of m. This design constraint ensures that the output can be seamlessly reshaped into the final forecast horizon without information loss or interpolation, thereby eliminating the need for additional postprocessing.

4 Experiments

This section presents the experimental evaluation of the performance and efficiency of the proposed SegTSF model. First, the forecasting accuracy of SegTSF was compared with that of complex deep-learning and recent lightweight baselines. Next, computational efficiency was quantified based on the number of parameters, multiply–accumulate operations (MACs), inference/training time, and peak memory usage. Finally, ablation studies were performed to analyze the contributions of the inner period learning module and intra-/inter-segment learning stages.

4.1 Experimental Setup

To compare the proposed model directly with existing baselines, widely adopted benchmark datasets and standardized evaluation metrics were used.

4.1.1 Datasets

We used five standard benchmark datasets for time-series forecasting: ETTh1, ETTh2, Electricity, Weather, and Traffic, as presented in Table 2. These datasets were adopted because of their distinct periodicities and characteristics, which are ideal for assessing the generalization performance of a model.

- ETTh1 and ETTh2 contain hourly measurements of seven variables from power transformers. ETTh1 exhibits greater variability than ETTh2 and possesses pronounced periodicity. Conversely, ETTh2 exhibits milder variability than ETTh1 and somewhat weaker periodicity.
- Electricity records the hourly electricity consumption of 321 customers. As consumption patterns vary across customers, these records are used to evaluate how well a model learns complex and heterogeneous patterns.
- Weather contains meteorological measurements recorded at 10-min intervals across 21 variables, including temperature, humidity, and wind speed. Its high sampling frequency and clear daily periodicity make it suitable for evaluating short-term forecasting under fine-grained temporal resolution.
- Traffic records hourly road occupancy rates from 862 sensors on freeways in the San Francisco Bay area. With a large number of heterogeneous channels, this dataset is used to assess model robustness under complex and diverse multi-channel patterns.

Table 2: Dataset summary.

Dataset	ETTh1	ETTh2	Electricity	Weather	Traffic
Channels	7	7	321	21	862
Frequency	1 h	1 h	1 h	10 min	1 h
Timesteps	17,420	17,420	26,304	52,696	17,544

4.1.2 Baselines

To evaluate the performance of the proposed SegTSF, two groups of baselines were selected: complex and lightweight models. For complex models, Informer, Autoformer, and PatchTST, a Transformer-based model with strong performance, were selected; for lightweight models, DLinear, FITS, and SparseTSF were selected.

4.1.3 Evaluation Metrics

The mean squared error (MSE) and mean absolute error (MAE), which are standard metrics in time-series forecasting, were used to evaluate forecasting accuracy. Additionally, parameters, multiply–accumulate

operations (MACs), inference time, epoch time, total training time, and maximum memory were used to evaluate model efficiency.

- MSE: Average of the squared prediction errors.
- MAE: Average of the absolute prediction errors. MAE provides a more interpretable and robust measure of average prediction deviation that is less sensitive to outliers than MSE.

To evaluate the stability of the reported results, SegTSF was evaluated over multiple runs with different random seeds, and the mean and standard deviation are reported.

- Parameters: Number of trainable parameters in the model.
- MACs: Computational complexity measured as the number of multiply accumulate operations per forward pass.
- Inference Time: Time required to produce outputs at test time for a single batch.
- Epoch Time: Time required to complete one training epoch, reported as the mean epoch time observed during model training.
- Total Training Time: Total time required to complete the full training process, including all epochs and evaluation steps.
- Max Memory: Peak GPU/system memory usage observed during training.
- Unless otherwise noted, lower values indicate better performance for all metrics.

4.1.4 Experimental Environment

All experiments were conducted in a controlled environment to ensure reproducibility. The hardware was an NVIDIA GeForce RTX 3060 Ti, and the software stack consisted of Python 3.8.20 and PyTorch 2.4.1, running in a Docker 4.36.0 container on Ubuntu 24.04.3 Long Term Support (LTS) for training and evaluation of all models. The datasets were split into training, validation, and test sets following established conventions for each dataset. For the ETT datasets (ETTh1 and ETTh2), we applied a 6:2:2 split ratio. For the remaining datasets (Electricity, Weather, and Traffic), we applied a 7:1:2 split ratio. For accuracy evaluation, all baseline models were reproduced by retraining each model using its official implementation under the experimental environment described above. Each baseline was configured with its own best-validated hyperparameters as reported in the respective original papers, as listed in Table 3. Using each model's optimal configuration ensures that every baseline is evaluated at its best performance potential, which represents a more meaningful comparison than forcing a unified hyperparameter setting that would disadvantage models sensitive to their specific configurations. The forecasting horizons $\{96, 192, 336, 720\}$ and hardware environment were kept identical across all models. The lookback window length $L = 720$ was selected to ensure adequate temporal coverage for both period-level and segment-level learning. With period $w = 24$, the model observes $n = 30$ complete cycles, providing sufficient temporal context for reliable forecasting across all horizons. A shorter lookback window such as $L = 96$ would reduce this to only $n = 4$ cycles, and would result in an insufficient number of time steps for segment-level learning. Finally, for the ablation study, the same hyperparameters as those used for accuracy evaluation were applied.

Table 3: Common hyperparameters used for accuracy evaluation.

Model	Lookback Window	Learning Rate	Batch Size	Epochs	Patience	Others
Informer	96	0.0001	32	10	3	label_len = 48
Autoformer	336	0.0001	32	10	3	label_len = 336
PatchTST	336	0.0001	128	30	5	patch_len = 16, stride = 8
DLinear	336	0.005	32	30	5	kernel_size = 25
FITS	720	0.0005	64	30	5	cutoff K = 30
SparseTSF	720	0.02	256	30	5	Period = 24
SegTSF (Ours)	720	0.02	256	30	5	Period = 24

4.2 Results

4.2.1 Performance Evaluation

Table 4 provides a comprehensive evaluation of the multivariate time-series forecasting performance across various datasets and horizons. As the bold and underlined values indicate the best and second-best performance, respectively, the proposed SegTSF demonstrates a decisive competitive advantage in the primary target horizons ($H = 96$ and 192). The standard deviations of SegTSF over multiple runs with different random seeds are consistently below 0.01, demonstrating the stability of the reported results. Note that repeated-run variance was measured only for the proposed model.

Table 4: Forecasting performance comparison on benchmark datasets. Both MSE and MAE are reported for each model.

Category		Linears								Transformers					
Method		SegTSF (Ours)		SparseTSF		FITS		DLinear		PatchTST		Autoformer		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.349 ± 0.001**	**0.383 ± 0.001**	0.362	0.389	0.375	0.405	0.375	0.399	0.370	0.400	0.435	0.446	0.941	0.769
	192	**0.393 ± 0.001**	**0.406 ± 0.001**	0.403	0.412	0.408	0.425	0.405	0.416	0.413	0.429	0.456	0.457	1.007	0.786
	336	**0.413 ± 0.004**	0.433 ± 0.006	0.434	**0.428**	0.429	0.442	0.439	0.443	0.422	0.440	0.486	0.487	1.038	0.784
	720	**0.418 ± 0.001**	**0.443 ± 0.004**	0.426	0.448	0.427	0.455	0.472	0.490	0.447	0.468	0.515	0.517	1.144	0.857
ETTh2	96	**0.271 ± 0.002**	**0.332 ± 0.001**	0.294	0.346	0.274	0.336	0.289	0.353	0.274	0.337	0.332	0.368	1.549	0.952
	192	**0.328 ± 0.001**	**0.371 ± 0.001**	0.339	0.377	0.333	0.375	0.383	0.418	0.341	0.382	0.426	0.434	3.792	1.542
	336	0.356 ± 0.002	0.396 ± 0.003	0.359	0.398	0.340	0.396	0.484	0.465	**0.329**	**0.384**	0.477	0.479	4.215	1.642
	720	0.379 ± 0.001	**0.422 ± 0.001**	0.383	0.425	**0.374**	0.423	0.605	0.551	0.379	**0.422**	0.453	0.490	3.656	1.619
Electricity	96	0.138 ± 0.001	0.234 ± 0.001	0.138	0.234	0.138	0.248	0.140	0.237	**0.129**	**0.222**	0.196	0.313	0.304	0.393
	192	0.151 ± 0.001	0.245 ± 0.001	0.154	0.245	0.152	0.260	0.153	0.249	**0.147**	**0.240**	0.211	0.324	0.327	0.417
	336	0.168 ± 0.003	0.263 ± 0.004	0.170	0.260	0.166	0.275	0.169	0.267	**0.163**	**0.259**	0.214	0.327	0.333	0.422
	720	0.208 ± 0.003	0.296 ± 0.004	0.209	0.297	0.205	0.305	0.203	0.301	**0.197**	**0.290**	0.236	0.342	0.351	0.427
Weather	96	0.168 ± 0.001	0.222 ± 0.001	0.169	0.223	0.170	0.225	0.176	0.239	**0.149**	**0.198**	0.249	0.329	0.354	0.405
	192	0.211 ± 0.001	0.258 ± 0.001	0.214	0.262	0.212	0.260	0.220	0.282	**0.194**	**0.241**	0.325	0.370	0.419	0.434
	336	0.257 ± 0.001	0.294 ± 0.001	0.257	0.294	0.258	0.294	0.265	0.319	**0.245**	**0.282**	0.351	0.391	0.583	0.543
	720	0.321 ± 0.001	0.339 ± 0.001	0.322	0.340	0.320	0.339	0.323	0.362	**0.314**	**0.334**	0.415	0.426	0.916	0.705
Traffic	96	0.388 ± 0.001	0.266 ± 0.001	0.389	0.268	0.398	0.286	0.410	0.282	**0.360**	**0.249**	0.597	0.371	0.733	0.410
	192	0.399 ± 0.001	0.272 ± 0.001	0.399	0.270	0.409	0.289	0.423	0.287	**0.379**	**0.256**	0.607	0.382	0.777	0.435
	336	0.413 ± 0.001	0.277 ± 0.001	0.411	0.276	0.421	0.294	0.436	0.296	**0.392**	**0.264**	0.623	0.387	0.776	0.434
	720	0.449 ± 0.001	0.297 ± 0.001	0.449	0.297	0.457	0.311	0.466	0.315	**0.432**	**0.286**	0.639	0.395	0.827	0.466

On the ETTh1 dataset, SegTSF achieves the best MSE across all horizons. In terms of MAE, SegTSF records the best performance across most horizons, with only a marginal difference at $H = 336$ where SparseTSF records a slightly lower value. On the ETTh2 dataset, SegTSF achieves the best MSE and MAE at short-term horizons, outperforming all baselines including Transformer-based models. At longer horizons, performance is mixed; PatchTST achieves the best results at $H = 336$, while SegTSF records the best MAE at $H = 720$ among all models. This empirical success on ETTh1 and ETTh2 confirms that the hierarchical segment-wise architecture of SegTSF effectively captures volatile short-term trends that are often smoothed out by conventional global mapping methods.

On the Electricity, Weather, and Traffic datasets, PatchTST maintains the best overall performance. Among linear-based lightweight models, SegTSF achieves the best or tied-best performance at short-term horizons on Electricity and Weather in both MSE and MAE. On Traffic, which contains 862 heterogeneous channels, SegTSF maintains competitive accuracy among lightweight linear models across all horizons. The relatively limited gains on these datasets are attributed to the structural challenge posed by complex and heterogeneous channel patterns under a channel-independent framework. Nevertheless, SegTSF consistently demonstrates robust performance among lightweight linear models across all three datasets.

Regarding longer horizons ($H = 336$ and 720), although competing models occasionally exhibit lower errors on certain datasets, this is considered a deliberate architectural trade-off. By specifically designing the model to prevent the 'averaging effect' and prioritize fine-grained local dynamics, SegTSF excels in the critical short-term window at the cost of reduced sensitivity to long-term periodic drift. This strategic focus aligns with the primary objective of providing highly efficient and accurate forecasts within a lightweight architecture.

To further investigate the fairness of the comparison, an additional controlled experiment was conducted in which all baselines were evaluated under a matched lookback window of $L = 720$, as presented in Table 5, where bold and underlined values indicate the best and second-best performance, respectively. The results show that most baselines exhibited degraded performance relative to their own best-validated configurations under this unified setting, confirming that each model has its own optimal lookback window. This degradation is not uniform across datasets, reflecting the dataset-specific nature of optimal lookback choices. This supports the choice to evaluate each model under its best-validated configuration in Table 4, which represents a fairer comparison of each model's true performance potential.

4.2.2 Efficiency Analysis

Table 6 summarizes the efficiency comparison across models using multiple metrics, where bold and underlined values indicate the best and second-best performance, respectively. All measurements were obtained on ETTh1 with the forecasting horizon set to 720. SegTSF uses only 0.66 K parameters, which is approximately 28% fewer than SparseTSF, which is the smallest baseline. It also exhibited a large margin over PatchTST in terms of computation and memory usage. The inference time, epoch time, and total training time were the lowest among all methods, and both MACs and peak memory were comparable to or better than those of the competing models. Overall, SegTSF substantially reduces the parameter budget while improving forecasting accuracy. These results suggest that the proposed architecture has the potential to be beneficial in environments where computational efficiency is a priority, though further validation in diverse deployment settings remains as future work.

Table 5: Forecasting performance comparison under a matched lookback window of $L = 720$ for all models.

Category		Linears								Transformers					
Method		SegTSF (Ours)		SparseTSF		FITS		DLinear		PatchTST		Autoformer		Informer	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	**0.349 ± 0.001**	**0.383 ± 0.001**	0.362	0.389	0.375	0.405	0.381	0.407	0.375	0.404	0.594	0.584	1.269	0.855
	192	**0.393 ± 0.001**	**0.406 ± 0.001**	0.403	0.412	0.408	0.425	0.419	0.430	0.418	0.430	0.606	0.588	1.487	0.943
	336	**0.413 ± 0.004**	**0.433 ± 0.006**	0.434	0.428	0.429	0.442	0.454	0.456	0.462	0.460	0.616	0.595	1.544	0.945
	720	**0.418 ± 0.001**	**0.443 ± 0.004**	0.426	0.448	0.427	0.455	0.499	0.512	0.495	0.497	0.742	0.660	1.481	0.975
ETTh2	96	**0.271 ± 0.002**	**0.332 ± 0.001**	0.294	0.346	0.274	0.336	0.302	0.367	0.290	0.354	0.471	0.487	5.189	1.812
	192	**0.328 ± 0.001**	**0.371 ± 0.001**	0.339	0.377	0.333	0.375	0.411	0.437	0.382	0.419	0.485	0.492	6.514	2.011
	336	0.356 ± 0.002	**0.396 ± 0.003**	0.359	0.398	**0.340**	**0.396**	0.542	0.514	0.421	0.448	0.474	0.488	5.284	1.859
	720	0.379 ± 0.001	**0.422 ± 0.001**	0.383	0.425	**0.374**	0.423	0.888	0.670	0.404	0.444	0.512	0.513	4.955	1.884
Electricity	96	0.138 ± 0.001	0.234 ± 0.001	0.138	0.234	0.138	0.248	0.136	0.236	**0.130**	**0.225**	0.208	0.323	0.395	0.460
	192	0.151 ± 0.001	0.245 ± 0.001	0.154	0.245	0.152	0.260	0.150	0.249	**0.147**	**0.241**	0.229	0.344	0.405	0.460
	336	0.168 ± 0.003	0.263 ± 0.004	0.170	0.260	0.166	0.275	0.166	0.266	**0.162**	**0.257**	0.239	0.358	0.404	0.460
	720	0.208 ± 0.003	0.296 ± 0.004	0.209	0.297	0.205	0.305	**0.200**	0.299	0.201	**0.293**	0.235	0.349	0.429	0.477
Weather	96	0.168 ± 0.001	0.222 ± 0.001	0.169	0.223	0.170	0.225	0.169	0.228	**0.149**	**0.201**	0.444	0.357	0.283	0.361
	192	0.211 ± 0.001	0.258 ± 0.001	0.214	0.262	0.212	0.260	0.214	0.272	**0.191**	**0.244**	0.574	0.368	0.445	0.461
	336	0.257 ± 0.001	0.294 ± 0.001	0.257	0.294	0.258	0.294	0.256	0.306	**0.245**	**0.288**	0.551	0.375	0.587	0.526
	720	0.321 ± 0.001	0.339 ± 0.001	0.322	0.340	0.320	0.339	**0.315**	0.354	**0.315**	**0.336**	0.597	0.389	0.953	0.703
Traffic	96	0.388 ± 0.001	0.266 ± 0.001	0.389	0.268	0.398	0.286	0.396	0.288	**0.370**	**0.257**	0.590	0.491	0.829	0.487
	192	0.399 ± 0.001	0.272 ± 0.001	0.399	0.270	0.409	0.289	0.406	0.290	**0.382**	**0.262**	0.604	0.564	0.902	0.525
	336	0.413 ± 0.001	0.277 ± 0.001	0.411	0.276	0.421	0.294	0.419	0.296	**0.395**	**0.269**	0.624	0.550	0.949	0.545
	720	0.449 ± 0.001	0.297 ± 0.001	0.449	0.297	0.457	0.311	0.459	0.319	**0.434**	**0.289**	0.634	0.566	1.430	0.793

Table 6: Model efficiency comparison on the ETTh1 dataset.

Model	Parameters	MACs	Inference Time	Epoch Time	Total Training Time	Max Memory
Informer	703.11 K	366.68 M	1.62 s	8.37 s	160.7 s	5570.49 MB
Autoformer	602.89 K	441.00 M	49.73 s	362.83 s	1368.3 s	13,341.52 MB
PatchTST	8.71 M	353.04 M	1.49 s	7.53 s	383.9 s	6248.38 MB
DLinear	1.04 M	7.28 M	0.48 s	0.79 s	47.8 s	109.28 MB
FITS	10.50 K	742.65 K	0.55 s	0.81 s	38.5 s	130.14 MB
SparseTSF	0.92 K	**277.2 K**	0.52 s	0.73 s	27.1 s	**72.83 MB**
SegTSF (Ours)	**0.66 K**	302.4 K	**0.45 s**	**0.70 s**	**23.2 s**	77.99 MB

The slightly higher MACs and maximum memory of SegTSF compared to some lightweight baselines, such as SparseTSF, are attributed to the model's core architecture, i.e., the hierarchical segment structure. Unlike conventional lightweight models that rely on simple linear mapping or downsampling, SegTSF employs a two-stage computational process. First, it encodes intra-segment local patterns and subsequently integrates inter-segment correlations. Although this hierarchical approach inevitably results in a marginal increase in computational load, it is a deliberate design trade-off intended to prevent the 'averaging effect' and preserve the fine-grained local dynamics essential for accurate forecasting. Nevertheless, SegTSF maintained optimal efficiency regarding parameter count and peak memory, while demonstrating superior performance in actual inference and training latency compared to competing models.

4.2.3 Ablation Study

The ablation study is designed to evaluate the contribution of each architectural module rather than the effect of hyperparameter tuning. Specifically, the intra-period learning module and the hierarchical segment learning module are the two core architectural components of SegTSF, and their individual contributions are quantified by removing each module independently. Table 7 presents the ablation study results on the ETTh1 and ETTh2 datasets to quantify the contribution of the proposed modules, where bold values indicate the best performance. The comparison settings are as follows: (i) full SegTSF; (ii) w/o period, which removes the intra-period relationship module; and (iii) w/o seg, which removes the segment-wise hierarchical module. The MSE was used as the evaluation metric. At horizons H = 96 and 192, the full model consistently attained a lower MSE than both ablated variants across both datasets; for ETTh1, it remained superior at all horizons. By contrast, for ETTh2 at H = 336 and 720, the gains are small or marginal; at H = 720, w/o seg is comparable to or slightly better than the full model. Overall, the improvements in MSE were concentrated at H = 96 and 192. This pattern can be attributed to the joint effect of the two modules in reducing large residuals. Intra-period relationship learning mitigates phase mismatch errors, while segment-wise hierarchical representation leverages local trend information to dampen spikes. Because the MSE assigns a squared penalty to large errors, these effects translate into notable score reductions. However, as the horizon increases, exogenous factors, periodic drift, and error accumulation become more influential, thereby diluting the benefits of the modules and reducing the observed differences.

Table 7: MSE results from the ETTh1/ETTh2 ablation study.

Dataset	ETTh1				ETTh2			
Horizon	**96**	**192**	**336**	**720**	**96**	**192**	**336**	**720**
SegTSF	**0.349**	**0.394**	**0.413**	**0.418**	**0.271**	**0.328**	**0.355**	0.379
w/o period	0.352	0.394	0.43	0.421	0.283	0.338	0.356	0.379
w/o segment	0.355	0.397	0.43	0.423	0.274	0.332	**0.355**	**0.378**

5 Conclusion

In this paper, SegTSF is introduced, a lightweight model designed to improve forecasting performance through hierarchical segment learning. This architecture combines an intra-period relationship-learning module with a hierarchical segment-learning structure. The goal of this design is to enhance the representational capacity of linear layers for higher accuracy while preserving the low computational cost of lightweight models.

Previous lightweight models are often constrained by a fundamental trade-off between preserving local patterns and maintaining computational efficiency. Existing approaches either filter high-frequency details or use a single global mapping, which fails to capture position-dependent dynamics. SegTSF was explicitly designed to resolve this limitation using its hierarchical learning architecture. The intra-segment learning stage operates on short, independent segments to preserve local, high-frequency patterns that are often lost by other methods. Subsequently, the inter-segment learning stage comprises these detailed local features in a coherent global forecast. The entire two-stage process was implemented using simple linear layers, enabling a significant enhancement in model expressiveness while adhering to a minimal parameter budget.

The experimental evaluation on standard benchmarks provided quantitative support for this architectural design. SegTSF achieved competitive accuracy among lightweight linear models, with superior performance particularly on ETTh1 and ETTh2 datasets. This result was achieved while requiring fewer

parameters than the other models. The combination of forecasting accuracy and computational efficiency indicates that SegTSF is a practical and efficient solution for multivariate time-series forecasting within a lightweight architecture.

Despite its strengths, SegTSF has several limitations that should be acknowledged. First, the model relies on a single predefined period, which may lead to performance degradation when the period is misspecified or when multiple periodicities coexist in the data. Second, as the forecast horizon increases, the performance gains of SegTSF diminish relative to competing models, attributed to periodic drift and the influence of exogenous factors that cannot be captured by a fixed periodicity assumption. Third, while MSE and MAE are reported in this study, incorporating additional domain-relevant metrics in future evaluations would provide a more comprehensive assessment of forecasting performance. Finally, edge-device validation remains as future work, as the current study focuses on computational efficiency through parameter reduction rather than deployment on specific hardware platforms. Addressing these limitations will be the focus of future work.

Future work will focus on addressing these key limitations of the current architecture. Adaptive structures will be explored that can adjust the model's focus from local to global patterns to extend its effectiveness to long-term forecasting horizons. Modules for the automated detection and handling of multiple periodicities are also planned to be developedand handling of multiple periodicities, eliminating dependency on a single predefined period. Additionally, a comprehensive hyperparameter sensitivity analysis, including segment length, period choice, and output dimensions, will be conducted to provide deeper insights into the model's behavior.

Acknowledgement: Not applicable.

Funding Statement: The present research has been conducted by the Research Grant of Kwangwoon University in 2025. This research was also supported by Korea Institute for Advancement of Technology (KIAT) grant funded by the Korea Government (MOTIE) (RS-2024-00406796, HRD Program for Industrial Innovation) and supported by the National Research Foundation of Korea (NRF) grant funded by the Korea government (MSIT) (RS-2022-NR068754).

Author Contributions: The authors confirm contribution to the paper as follows: Conceptualization, Hyunjun Park, Dong-Hyuk Im; methodology, Hyunjun Park; software, Hyunjun Park; validation, Hee-Gook Jun, Seongyong Kim, Dong-Hyuk Im; investigation, Hyunjun Park, Dong-Hyuk Im; resources, Dong-Hyuk Im; writing—original draft preparation, Hyunjun Park; writing—review and editing, Hee-Gook Jun, Seongyong Kim, Dong-Hyuk Im; visualization, Hyunjun Park; supervision, Dong-Hyuk Im. All authors reviewed and approved the final version of the manuscript.

Availability of Data and Materials: The datasets used in this study are publicly available. ETTh1 and ETTh2 are available at https://github.com/zhouhaoyi/ETDataset. The Electricity, Weather, and Traffic datasets are available at https://github.com/thuml/Time-Series-Library. The source code will be made publicly available upon acceptance of the paper.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Wen Q, Zhou T, Zhang C, Chen W, Ma Z, Yan J, et al. Transformers in time series: a survey. In: Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence; 2023 Aug 19–25; Macau, China. doi:10.24963/ijcai.2023/759.

2. Wang X, Han Y, Leung VCM, Niyato D, Yan X, Chen X. Convergence of edge computing and deep learning: a comprehensive survey. IEEE Commun Surv Tutor. 2020;22(2):869–904. doi:10.1109/COMST.2020.2970550.
3. Pinheiro MG, Madeira SC, Francisco AP. Short-term electricity load forecasting—a systematic approach from system level to secondary substations. Appl Energy. 2023;332:120493. doi:10.1016/j.apenergy.2022.120493.
4. Chen S, Chen X, Li Y. Memory-augmented short time series forecasting. In: Database systems for advanced applications. Singapore, Singapore: Springer; 2025.
5. Zeng A, Chen M, Zhang L, Xu Q. Are transformers effective for time series forecasting? Proc AAAI Conf Artif Intell. 2023;37(9):11121–8. doi:10.1609/aaai.v37i9.26317.
6. Xu Z, Zeng A, Xu Q. FITS: modeling time series with 10k parameters. In: Proceedings of the 12th International Conference on Learning Representations (ICLR 2024); 2024 May 7–11; Vienna, Austria. doi:10.48550/arXiv.2307.03756.
7. Lin S, Lin W, Wu W, Chen H, Yang J. SparseTSF: modeling long-term time series forecasting with 1k parameters. In: Proceedings of the 41st International Conference on Machine Learning; 2024 Jul 21–27; Vienna, Austria.
8. Tian G, Yang Y, Wen S. Time-series stock price forecasting based on neural networks: a comprehensive survey. Artif Intell Sci Eng. 2025;1(4):255–77. doi:10.23919/AISE.2025.000018.
9. Zhang GP. Time series forecasting using a hybrid ARIMA and neural network model. Neurocomputing. 2003;50:159–75. doi:10.1016/S0925-2312(01)00702-0.
10. Hippert HS, Pedreira CE, Souza RC. Neural networks for short-term load forecasting: a review and evaluation. IEEE Trans Power Syst. 2001;16(1):44–55. doi:10.1109/59.910780.
11. Salinas D, Flunkert V, Gasthaus J, Januschowski T. DeepAR: probabilistic forecasting with autoregressive recurrent networks. Int J Forecast. 2020;36(3):1181–91. doi:10.1016/j.ijforecast.2019.07.001.
12. Lai G, Chang WC, Yang Y, Liu H. Modeling long- and short-term temporal patterns with deep neural networks. In: Proceedings of the 41st International ACM SIGIR Conference on Research & Development in Information Retrieval; 2018 Jul 8–12; Ann Arbor, MI, USA. doi:10.1145/3209978.3210006.
13. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez AN, et al. Attention is all you need. In: Proceedings of the 31st International Conference on Neural Information Processing Systems; 2017 Dec 4–9; Long Beach, CA, USA.
14. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, et al. Informer: beyond efficient transformer for long sequence time-series forecasting. Proc AAAI Conf Artif Intell. 2021;35(12):11106–15. doi:10.1609/aaai.v35i12.17325.
15. Wu H, Xu J, Wang J, Long M. Autoformer: decomposition transformers with{auto-correlation}for long-term series forecasting. In: Proceedings of the 35th International Conference on Neural Information Processing Systems; 2021 Dec 6–14; Online.
16. Nie Y, Nguyen NH, Sinthong P, Kalagnanam J. A time series is worth 64 words: long-term forecasting with transformers. In: Proceedings of the 11th International Conference on Learning Representations; 2023 May 1–5; Kigali, Rwanda.
17. Das A, Kong W, Leach A, Mathur S, Sen R, Yu R. Long-term forecasting with TiDE: time-series dense encoder. Trans Mach Learn Res. 2023. doi:10.48550/arXiv.2304.08424.
18. Li Z, Qi S, Li Y, Xu Z. Revisiting long-term time series forecasting: an investigation on linear mapping. arXiv:2305.10721. 2023.
19. Ekambaram V, Jati A, Nguyen N, Sinthong P, Kalagnanam J. TSMixer: lightweight MLP-mixer model for multivariate time series forecasting. In: Proceedings of the 29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining; 2023 Aug 6–10; Long Beach, CA, USA. doi:10.1145/3580305.3599533.
20. Zhang Y, Yan J. Crossformer: transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: Proceedings of the 11th International Conference on Learning Representations; 2023 May 1–5; Kigali, Rwanda.
21. Ben Taieb S, Bontempi G, Atiya AF, Sorjamaa A. A review and comparison of strategies for multi-step ahead time series forecasting based on the NN5 forecasting competition. Expert Syst Appl. 2012;39(8):7067–83. doi:10.1016/j.eswa.2012.01.039.

22. Han L, Ye HJ, Zhan DC. The capacity and robustness trade-off: revisiting the channel independent strategy for multivariate time series forecasting. IEEE Trans Knowl Data Eng. 2024;36(11):7129–42. doi:10.1109/TKDE.2024.3400008.
23. Grabocka J, Schilling N, Wistuba M, Schmidt-Thieme L. Learning time-series shapelets. In: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining; 2014 Aug 24–27; New York, NY, USA. doi:10.1145/2623330.2623613.