Integrating CNN And Mamba For Effective Time Series Forecasting

Junliang Tao, Hongbing Wang*, Li Cao, Chenhao Xie, CJian Li, and Liang Zhou

School of Big Data and Computer Science, Guizhou Normal University, Guiyang, 550025, China

Corresponding author. E-mail: hbwang@gznu.edu.cn

Received: Nov. 25, 2025; Accepted: Dec. 24, 2025

Recently, Transformer-based models have achieved remarkable progress in time series forecasting, yet they still suffer from limitations in efficiency and scalability due to quadratic complexity. In contrast, the newly proposed Mamba model demonstrates strong potential with its linear-time complexity and selective state space mechanism, making it more suitable for long-sequence modeling. Meanwhile, convolutional neural networks (CNNs) have long been recognized for their effectiveness in capturing local dependencies and fine-grained temporal patterns. To integrate the complementary advantages of these two paradigms, we propose PaDuM, a Patch-Based Dual-Stream Network that jointly leverages CNN and Mamba. Specifically, PaDuM first applies an exponential moving average (EMA) to adaptively decouple input sequences into trend and seasonal components, ensuring better interpretability and reduced noise. Each component is then patchified and modeled through dual streams, where CNN focuses on local seasonal variations while Mamba captures global trends with efficiency. To further enhance stability, we introduce a novel Sigmoid-based weight decay loss, which emphasizes recent predictions while preventing overfitting to distant horizons. Extensive experiments conducted on eight diverse real-world datasets spanning electricity consumption, traffic flow, and meteorological data consistently demonstrate that PaDuM achieves state-of-the-art forecasting performance, while maintaining strong robustness, scalability, and generalization ability across domains. The implementation and resources are publicly available at https://github.com/T-DXVN/PaDuM.

Keywords: Time Series Forecasting; State space model; CNN; Mamba; Patch-based Modeling

http://dx.doi.org/10.6180/jase.202608_31.045

1. Introduction

Time series forecasting (TSF) plays a vital role in a wide range of domains, including electricity load prediction [1], meteorological monitoring [2], energy management [3], traffic planning [4], and healthcare monitoring [5–10]. Accurate forecasting provides critical support for decision-making, resource allocation, and risk prevention, making it a long-standing re- search problem of both theoretical and practical significance.

With the rapid development of deep learning, Transformer-based models [11] have been widely adopted for TSF due to their powerful ability to model long-range dependencies through self- attention. Despite impressive performance, these models suffer from quadratic computational complexity with respect to sequence length and limited capacity in capturing local temporal patterns. These drawbacks severely hinder their scalability and accuracy, especially when handling very long input sequences. Consequently, achieving both high efficiency and high accuracy in long-horizon TSF re-mains a challenging and open problem.

To address the efficiency bottleneck, state space models (SSMs) [12–18], particularly Mamba [19], have recently emerged as promising alternatives.

Mamba enables linear-time sequence modeling and effectively captures global dependencies through its selective

memory mechanism. However, by primarily focusing on global trends, Mamba tends to overlook fine-grained short-term fluctuations and is less sensitive to high-frequency noise or abrupt local changes—limiting its ability to characterize detailed local semantics.

In contrast, convolutional neural networks (CNNs) excel at extracting short-term local patterns through sliding-window convolutions, offering strong responsiveness to high-frequency dynamics and sudden variations. Yet, their receptive fields are inherently limited by kernel size, making it difficult to model long-range temporal dependencies or capture crossscale global trends—especially in extended forecast-ing horizons.

This complementary nature motivates a natural synergy: combining the global, efficient modeling capability of Mamba with the local sensitivity of CNNs can jointly address the dual challenges of accuracy and efficiency in TSF. Moreover, raw time series often mix trend and seasonal components; directly applying patch-based representations—inspired by Vision Transformers (ViT) [20]—to such mixed signals may cause feature interference and degrade learning. To enable more structured modeling, we propose to first decouple the series into trend and seasonal components using exponential moving average (EMA) [21].

Building on these insights, we propose **PaDuM** (**P**atch-based **Du**al-stream Network with CNN and **M**amba), a novel architecture that processes the de- coupled seasonal and trend components through dedicated CNN and Mamba branches, respectively, and fuses them for robust forecasting. As shown in Fig. 1, extensive experiments on eight diverse real-world datasets consistently demonstrate that PaDuM outperforms state-of-the-art baselines in terms of both accuracy and robustness.

This work is an extended version of our conference submission to the 2026 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2026) [22]. Compared with the conference version, this paper provides more detailed model analysis, includes additional ablation studies, and presents a broader set of experiments to further validate the effectiveness and generalization of the proposed method.

Our main contributions are summarized as follows:

- To the best of our knowledge, this is the first work to introduce a hybrid framework that combines convolutional neural networks (CNNs) and the recently proposed Mamba state space model for time series forecasting. We propose **PaDuM**, *a patch-based dual-stream architecture* that jointly leverages the local feature extraction capability of CNNs and the efficient long-sequence modeling ability of Mamba. PaDuM achieves a balanced design that captures both short-term seasonal variations and long-term trend dependencies.

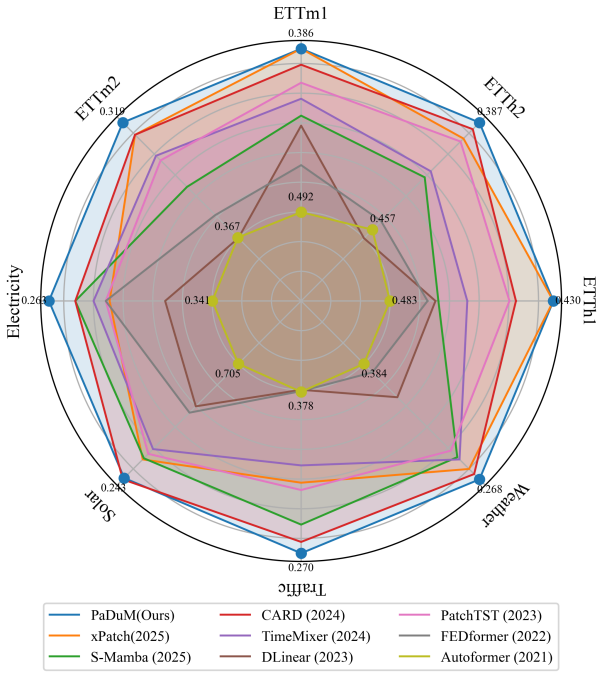

Fig. 1. Average MAE comparison between **PaDuM** and state-of-the-art baselines with a look-back window size of L = 96. In the radar chart, values closer to the outer boundary indicate better performance.

- We design a novel **Sigmoid-based loss function**, which adaptively emphasizes recent predictions while suppressing noise in distant horizons. This formulation not only improves forecasting ac- curacy but also enhances training stability, effectively addressing the common issue of error accumulation in long-term prediction tasks.

- We conduct comprehensive experiments on **eight real-world benchmark** datasets covering electricity, traffic, and meteorology domains. The results demonstrate that PaDuM consistently outperforms state-of-the-art baselines across multiple metrics, highlighting its robustness, scalability, and generalization ability in diverse forecasting scenarios.

2. Related work

In this section, we review the recent advancements in time series forecasting, focusing on four key areas: time series decomposition, patch-based modeling, depthwise separable convolution, and the application of Mamba in time

series. We discuss how these techniques address challenges such as capturing local and global dependencies, improving computational efficiency, and enhancing model interpretability. This overview provides the foundation for understanding the design choices and innovations in our proposed PaDuM framework.

2.1. Time Series Decoupling

In recent years, decomposition-based architectures have become a fundamental paradigm in time series forecasting, as they enable models to capture heterogeneous patterns by explicitly separating trend and seasonal components. For instance, Simple Moving Average (SMA) has been adopted by models such as Autoformer [23], FEDformer [24], and DLinear [25] for trend extraction, where a smoothed representation is obtained by averaging over fixed-length windows. Despite its simplicity, SMA inevitably suffers from strong lag effects and a reliance on bound- ary padding, which often leads to inaccurate trend estimation and reduced forecasting reliability. To alleviate these issues, xPatch [26] was the first to introduce Exponential Moving Average (EMA) for trend-seasonal decoupling. By assigning exponentially decreasing weights to past observations, EMA exhibits higher responsiveness to recent data, thereby improving both timeliness and robustness in forecasting. Inspired by this advancement, we adopt EMA to decouple raw time series into trend and seasonal components, and further incorporate this design into our proposed PaDuM framework, where decoupling serves as a crucial preprocessing step to enhance the discriminability of learned representations.

2.2. Patch-Based Time Series Modeling

Inspired by the breakthroughs of Vision Transformers (ViT) [20] in computer vision, patch-based modeling has recently been introduced into time series forecasting as an effective way to bridge local temporal patterns and global contextual relationships. PatchTST [27] applies a channel-independent patch segmentation strategy, where input sequences are divided into smaller patches that are processed individually. This design not only reduces sequence length and suppresses noise but also significantly improves long-term forecasting performance by capturing high-level temporal abstractions. Building on this idea, Crossformer [28] further employs crossdimension dynamic attention to jointly exploit correlations across both temporal and variable dimensions, thereby enhancing efficiency in long-sequence modeling. These works consistently validate the effectiveness of patching in time series and motivate our adoption of patch-based techniques in PaDuM. In particular, we extend patching to a dual-stream setting, where decoupled trend and seasonal components are patchified and modeled through complementary CNN and Mamba branches.

2.3. Depthwise Separable Convolution and Patch- Based Modeling

Depthwise Separable Convolution (DWConv) [29], originally proposed by Sifre and Mallat, decomposes standard convolution into two lightweight operations: a depthwise convolution applied independently on each channel, followed by a pointwise convolution for channel mixing. This design dramatically reduces both computational cost and parameter count, while retaining strong representational power. DWConv has been widely adopted in computer vision, with successful implementations in Inception networks [30, 31], MobileNet [32], and Xception [33], where efficiency is critical for large-scale or resourceconstrained applications. More recently, ConvMixer [34] employed DWConv in a patch-based context to effectively process image patches, demonstrating its advantages in capturing local features under patchlevel representations. Inspired by these findings, our PaDuM model employs CNN with depthwise separable convolution to extract fine-grained seasonal patterns, thus achieving both efficiency and accuracy in the seasonal stream.

2.4. Applications of Mamba in Time Series

Mamba has recently emerged as a promising paradigm for efficient and scalable time series forecasting, owing to its selective state mechanism and linear computational complexity. Unlike Transformers, which rely on quadratic self-attention, Mamba maintains efficiency by updating latent states selectively, enabling long-sequence modeling with significantly reduced cost. Several studies have validated its applicability in multivariate forecasting tasks. For example, S-Mamba [35] integrates bidirectional Mamba with feed-forward networks to capture inter-variable correlations, effectively balancing accuracy with computational efficiency. TimeMachine [36] explores a fully state-space-model-based architecture, incorporating multi-scale downsampling where Mamba modules jointly capture both global and local temporal contexts. These studies collectively highlight Mamba's ability to handle multi-scale dependencies and to generalize well across domains. Building upon this line of research, we integrate Mamba into PaDuM as the trend modeling branch, allowing it to efficiently capture long-range dependencies in the trend component, while the CNN branch complements it by focusing on seasonal details.

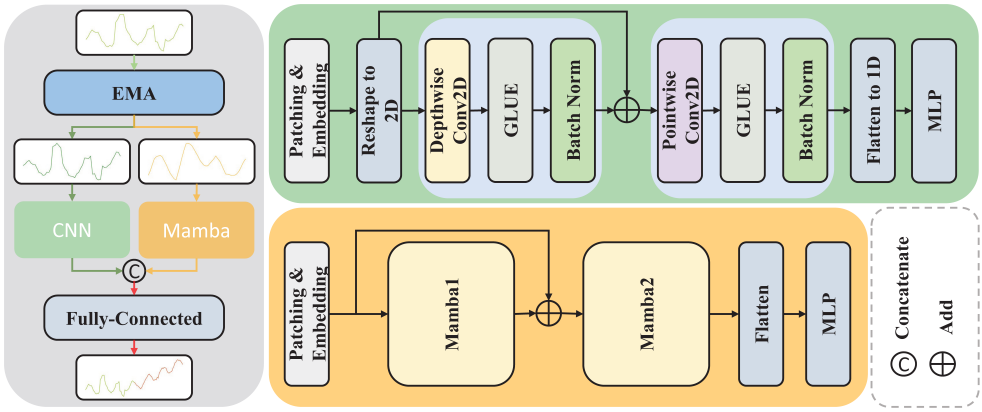

Fig. 2. Overview of the proposed **PaDuM** framework. The input time series is decomposed into seasonal and trend components, which are modeled by CNN and Mamba streams, respectively. CNN extracts local and periodic patterns, while Mamba, as a state space model (SSM), captures long-term dependencies. Their outputs are fused via a fully connected (FC) layer to generate the final prediction.

3. Methods

In this section, we introduce the proposed PaDuM framework, which is designed for efficient and accurate multivariate time series forecasting. The framework begins with an EMA-based decomposition to separate trend and seasonal components, followed by a dual-stream architecture: a CNN stream for capturing local seasonal patterns and a Mamba stream for modeling long-range trends. The outputs of the two streams are fused to generate final predictions. Additionally, we propose a Sigmoid-based loss function to enhance training stability and improve forecasting accuracy. The detailed methodology is presented in the following subsections.

3.1. Problem Statement

Given a multivariate time series $\mathrm{X} \in \mathbf{R}^{L\times C}$, where L denotes the length of the historical observations (i.e., input sequence length) and C is the number of variables (channels), the goal of the time series forecasting (TSF) task is to predict the future sequence $\mathrm{Y} \in \mathbf{R}^{T\times C}$ over the next T time steps based on the past L observations.

Formally, the task aims to learn a mapping function:

$$f : \mathbf{R}^{L\times C} \to \mathbf{R}^{T\times C} \tag{1}$$

such that the predicted sequence $\hat{\mathrm{Y}} = f(\mathrm{X})$ approximates the true future sequence Y as closely as possible.

3.2. Framework

As illustrated in Fig. 2, the core idea of **PaDuM** is to decompose the time series into two sub-streams seasonal and trend - which are modeled by CNN and Mamba, respectively. CNN excels at capturing local patterns and periodic features, making it suitable for modeling seasonal variations in time series; meanwhile, Mamba, as a state space model (SSM), can efficiently capture long-term dependencies, making it well-suited for trend modeling. Finally, the outputs of the two branches are fused through a fully connected (FC) layer to generate the final prediction.

The model input is a time series $x \in \mathbf{R}^{B\times L\times C}$, where B denotes the batch size, L is the input sequence length, and C is the number of channels. The output is a predicted sequence $y \in \mathbf{R}^{B\times T\times C}$, where T denotes the forecasting horizon. The overall process of PaDuM can be formulated as:

$$y = \mathrm{PaDuM}\left(x_s, x_t\right) \tag{2}$$

$$= \mathrm{FC}\ \mathrm{Concat}\ f_{\mathrm{CNN}}\left(x_s\right),\ f_{\mathrm{Mamba}}\left(x_t\right). \tag{3}$$

3.3. CNN Stream

Patching: The input sequence $x \in \mathbf{R}^{B\times C\times L}$ is reshaped to $\mathbf{R}^{B\cdot C\times L}$ and divided into multiple subsequences of length P with stride S using the unfold operation. The patched tensor has the shape: $x_p \in \mathbf{R}^{B\cdot C\times N\times P}$ where $N = \lfloor (L - P)/S \rfloor + 1$ denotes the number of patches.

Patch Embedding: Each patch is projected into a high-dimensional feature space via a linear layer:

$$x_e = \mathrm{GELU}\,\mathrm{BN1d}\left(\mathrm{FC1}\left(x_p\right)\right) \in \mathbf{R}^{B\cdot C\times N\times D} \tag{4}$$

where $D = P \cdot P$ is the embedding dimension, FC1 is a linear layer, BN1d denotes batch normalization, and GELU is the activation function.

Depthwise Convolution: The patch embeddings are reshaped into a 2D tensor $x_e \in \mathbf{R}^{B\cdot C\times N\times P\times P}$, followed by depthwise convolution (Depthwise Conv2D) to capture local patterns:

$$x_d = \text{GELU BN2d } (\text{DWConv2d}\,(x_e)) + \text{FC2}\,(x_e) \quad (5)$$

where FC2 is a projection layer for residual connection, ensuring dimensional consistency.

Pointwise Convolution: A pointwise convolution (Pointwise Conv2D) is applied to further aggregate features, while reducing the number of channels by a factor of t:

$$x_c = \text{GELU BN 2 d}\,(\text{PTConv 2 d}\,(x_d)) \in \mathbf{R}^{B\cdot C\times(N/t)\times P\times P} \quad (6)$$

Flattening and Prediction: The features are flattened and passed through a two-layer fully connected network to generate the prediction:

$$\begin{aligned} x_f &= \text{Flatten}\,(x_c) \in \mathbf{R}^{B\cdot C\times(N/t)\cdot P\cdot P}, \\ y_s &= \text{FC4GELU}\left(\text{FC 3}\left(x_f\right)\right) \in \mathbf{R}^{B\cdot C\times T}, \\ y_s &= \text{Reshape}\,(y_s) \in \mathbf{R}^{B\times T\times C}. \end{aligned} \quad (7)$$

3.4. Mamba Stream

Patching: Similar to the CNN Stream, the input sequence is divided into patches, resulting in $x_p \in \mathbf{R}^{B\cdot C\times N\times P}$.

Patch Embedding: Each patch is mapped into a latent space of dimension D via a linear layer:

$$x_e = \text{GELU BN 1 d}\,(\text{FC 1}\,(x_p)) \in \mathbf{R}^{B\cdot C\times N\times D} \quad (8)$$

Mamba Layer: Two stacked Mamba modules are applied to the patch embeddings to capture longrange dependencies:

$$x_m = \text{Mamba 1}\,(x_e) + \text{FC 2}\,(x_e) \quad (9)$$

$$x_m = \text{Mamba 2}\,(x_m) \quad (10)$$

where Mamba1 and Mamba2 denote Mamba blocks, and FC2 serves as a residual projection layer.

Flattening and Prediction: The processed features are flattened and passed through fully connected layers to generate predictions:

$$\begin{aligned} x_f &= \text{Flatten}\,(x_m) \in \mathbf{R}^{B\cdot C\times N\cdot D}, \\ y_t &= \text{FC 4 GELU}\left(\text{FC 3}\left(x_f\right)\right) \in \mathbf{R}^{B\cdot C\times T}, \\ y_t &= \text{Reshape}\,(y_t) \in \mathbf{R}^{B\times T\times C}. \end{aligned} \quad (11)$$

Algorithm 1. Comparison of time performance and performance with different superpixel methods

Require: Input time series $x \in \mathbf{R}^{B\times L\times C}$, patch length P, stride S, horizon T
Ensure: Prediction $\hat{y} \in \mathbf{R}^{B\times T\times C}$
1: Decompose input: $x \to (x_s, x_t)$ **CNN Stream (Seasonal)**
2: $x_p^s \leftarrow \text{Unfold}\,(x_s, P, S)$
3: $y_s \leftarrow f_{\text{CNN}}\left(x_p^s\right)$ **Mamba Stream (Trend)**
4: $x_p^t \leftarrow \text{Unfold}\,(x_t, P, S)$
5: $y_t \leftarrow f_{\text{Mamba}}\left({x^t}_p\right)$ **Fusion**
6: $\hat{y} \leftarrow \text{FC}\,(\text{Concat}\,(y_s, y_t))$ **return** $\hat{y}$

3.5. Fusion Layer

The outputs of the seasonal and trend streams, $y_s, y_t \in \mathbf{R}^{B\times T\times C}$, are fused via a concatenation operation:

$$x_{\text{cat}} = \text{Concat}\,(y_s, y_t) \in \mathbf{R}^{B\times 2T'\times C} \quad (12)$$

Finally, a fully connected layer maps the fused features into the final prediction space:

$$\hat{y} = \text{FC}\,(x_{\text{cat}}) \in \mathbf{R}^{B\times T\times C} \quad (13)$$

For clarity, the pseudocode of PaDuM is provided in Algorithm 1 to illustrate its overall workflow.

3.6. Sigmoid Loss

During training, we observed that the loss functions adopted in CARD [19] and xPatch [8] were not wellsuited for our model optimization. Therefore, we designed a **Sigmoid-based weighted decay loss function**. The key idea is to assign higher importance to near-term predictions, thereby reducing the difficulty of learning long-term predictions and enabling the model to gradually improve its performance.

The loss function is defined as:

$$L_{\text{sigmoid}} = \frac{1}{T}\sum_{i=1}^{T}\rho_{\text{sigmoid}}\,(i)\left\|\hat{x}_{1:T}^{(i)} - x_{1:T}^{(i)}\right\| \quad (14)$$

where T denotes the prediction horizon (the number of future time steps). $\hat{x}_{1:T}(i)$: predicted values of the i th sample over time steps $1\ldots T x_{1:T}^{(i)}$: ground-truth observations of the i-th sample over time steps $1\ldots T$.

The weighting coefficient $\rho_{\text{sigmoid}}\,(i)$ is formulated as:

$$\rho_{\text{sigmoid}}\,(i) = l + \frac{1-l}{1+\exp(k\cdot(i-c))} \quad (15)$$

where:l controls the minimum weight assigned to long-term predictions, k determines the decay rate of the weighting curve, c specifies the inflection point that separates "near-term" from "long-term" predictions.

This design ensures that the model places greater emphasis on short-term forecasting at the early stage of training while still gradually learning to capture long-term dependencies.

4. Results and discussion

In this section, we evaluate the performance of the proposed PaDuM framework through extensive experiments on eight benchmark datasets. We compare PaDuM against state-of-the-art models across various metrics, conduct ablation studies to analyze the impact of key design choices, and assess the model's training efficiency and effectiveness. Additionally, we visualize the forecasting results to demonstrate PaDuM's ability to capture both global trends and local patterns. The results highlight the robustness, scalability, and superior performance of PaDuM in diverse time series forecasting scenarios.

4.1. Datasets

We evaluated the performance of our PaDuM model on eight widely used multivariate time series datasets, including four ETT variants (ETTh1, ETTh2, ETTm1, ETTm2), Electricity, Solar-Energy, Traffic, and Weather. The detailed statistics of these benchmark datasets are summarized in Table 2.

- **ETT**[1]: Contains two hourly (ETTh1, ETTh2) and two $15-$min (ETTm1, ETTm2) electricity load datasets from July 2016 to July 2018. The period windows are 24 for hourly and 96 for $15-$min data, suitable for evaluating both short- and long-term dependencies.

- **Electricity** [2]: Hourly electricity consumption of 321 customers from 2012 to 2014, with a daily cycle (24). The dataset exhibits multi-periodic patterns influenced by external factors such as weather and is suitable for multi-scale forecasting.

- **Solar-Energy** [3]: Solar power generation from 137 PV plants in 2006, sampled every 10 min (period 144). It shows daily periodicity with additional variability from weather and holidays.

- **Traffic** [4]: Hourly road occupancy rates collected by 862 sensors in the San Francisco Bay area from 2015 to 2016, reflecting traffic flow patterns with strong temporal regularities.

- **Weather** [5]: Meteorological observations from a German station in 2020, including 21 variables (e.g., temperature, humidity), sampled every 10 min (period 144), suitable for periodic feature extraction.

4.2. Evaluation metrics

Following prior work, we adopt Mean Squared Error (MSE) and Mean Absolute Error (MAE) as evaluation metrics.The formulas for these metrics are as follows:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \quad \mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_i - \hat{y}_i| \tag{16}$$

where y_i denotes the ground truth value of the i-th sample, $\hat{y}_i$ represents the corresponding predicted value, and N is the total number of samples. MSE (Mean Squared Error) penalizes larger errors more heavily due to the squaring operation, making it sensitive to outliers. MAE (Mean Absolute Error) measures the average magnitude of errors without considering their direction, providing a more robust evaluation of overall prediction accuracy.

4.3. Baseline

We evaluate PaDuM against representative models from the past five years based on different methodological paradigms: (1) Transformer-based models: CARD [39], PatchTST [27], FEDformer [24], and Autoformer [23]; (2) MLP-based models: TimeMixer [40] and DLinear [25]; (3) CNN-MLP hybrid models: xPatch [26]; and (4) Mamba-based models: S-Mamba [35]. All experiments are conducted under the Time Series Library (TS-Lib) [41]. The brief introductions of these models are as follows:

- CARD is a Transformer-based model using channel-aligned attention to capture temporal and variable dependencies, with a multi-scale token blend module and robust loss to enhance representation.

- xPatch is a dual-stream, non-transformer model that applies seasonal-trend exponential decomposition, combining an MLP linear stream and CNN nonlinear stream with robust loss and sigmoid learning rate adjustment.

- S-Mamba is a Mamba-based model that tokenizes each variable independently, uses a bidirectional Mamba layer for inter-variable correlations, and a feed-forward network for temporal dependencies.

[1] https://github.com/zhouhaoyi/ETDataset
[2] https://archive.ics.uci.edu/ml/datasets/ElectricityLoadDiagrams20112014
[3] https://www.nrel.gov/grid/solar-power-data.html
[4] https://pems.dot.ca.gov
[5] https://www.bgc-jena.mpg.de/wetter/

Table 1. Main results. The look-back window is set to $L = 96$, and the prediction horizons are $T \in \{96, 192, 336, 720\}$. The best performance is highlighted in red, and the second best is marked in lightblue with underlining.

Method		PaDuM (our)		xPatch (2025)		S-Mamba (2025)		CARD (2024)		TimeMixer (2024)		DLinear (2023)		PatchTST (2023)		FEDformer (2022)		Autoformer (2021)	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.374	0.389	0.374	0.391	0.388	0.406	0.387	0.398	0.385	0.401	0.383	0.396	0.386	0.400	0.379	0.416	0.472	0.465
	192	0.434	0.420	0.433	0.422	0.445	0.441	0.442	0.427	0.443	0.430	0.439	0.433	0.442	0.431	0.423	0.446	0.476	0.467
	336	0.472	0.436	0.486	0.443	0.490	0.465	0.486	0.448	0.513	0.470	0.492	0.467	0.478	0.446	0.455	0.466	0.497	0.482
	720	0.514	0.473	0.483	0.465	0.506	0.497	0.483	0.469	0.493	0.473	0.524	0.515	0.474	0.466	0.485	0.495	0.528	0.517
	Avg	0.448	0.430	0.444	0.430	0.457	0.452	0.449	0.435	0.458	0.444	0.460	0.453	0.445	0.436	0.436	0.456	0.493	0.483
ETTh2	96	0.287	0.330	0.285	0.332	0.297	0.349	0.289	0.338	0.289	0.340	0.329	0.384	0.291	0.340	0.336	0.383	0.378	0.412
	192	0.360	0.378	0.365	0.382	0.378	0.399	0.367	0.387	0.381	0.396	0.432	0.443	0.372	0.391	0.434	0.443	0.455	0.456
	336	0.401	0.412	0.405	0.417	0.425	0.435	0.379	0.404	0.427	0.431	0.446	0.455	0.376	0.407	0.470	0.472	0.475	0.479
	720	0.408	0.428	0.422	0.437	0.432	0.448	0.407	0.429	0.444	0.452	0.776	0.632	0.409	0.433	0.458	0.474	0.459	0.480
	Avg	0.364	0.387	0.369	0.392	0.383	0.408	0.360	0.389	0.385	0.405	0.496	0.479	0.362	0.393	0.424	0.443	0.442	0.457
ETTml	96	0.320	0.344	0.323	0.348	0.331	0.368	0.323	0.353	0.315	0.355	0.346	0.374	0.320	0.358	0.337	0.388	0.435	0.441
	192	0.372	0.374	0.375	0.374	0.378	0.393	0.370	0.377	0.385	0.395	0.383	0.393	0.362	0.381	0.376	0.413	0.565	0.503
	336	0.396	0.395	0.398	0.392	0.410	0.414	0.400	0.397	0.393	0.406	0.414	0.414	0.391	0.404	0.439	0.451	0.560	0.512
	720	0.459	0.429	0.466	0.430	0.474	0.451	0.469	0.435	0.453	0.443	0.480	0.457	0.451	0.436	0.479	0.471	0.572	0.511
	Avg	0.387	0.386	0.390	0.386	0.398	0.406	0.391	0.390	0.386	0.400	0.406	0.410	0.381	0.395	0.408	0.431	0.533	0.492
ETTm2	96	0.174	0.251	0.176	0.254	0.182	0.266	0.173	0.253	0.175	0.256	0.187	0.281	0.176	0.259	0.180	0.274	0.220	0.310
	192	0.240	0.296	0.240	0.297	0.252	0.313	0.242	0.299	0.240	0.302	0.272	0.349	0.241	0.304	0.248	0.319	0.271	0.335
	336	0.301	0.336	0.303	0.338	0.313	0.349	0.302	0.337	0.302	0.341	0.344	0.395	0.301	0.342	0.315	0.364	0.339	0.376
	720	0.396	0.392	0.400	0.394	0.413	0.405	0.396	0.393	0.395	0.401	0.442	0.447	0.403	0.401	0.427	0.422	0.463	0.447
	Avg	0.278	0.319	0.280	0.321	0.290	0.333	0.278	0.321	0.278	0.325	0.311	0.368	0.280	0.326	0.292	0.345	0.323	0.367
Electricity	96	0.152	0.238	0.184	0.259	0.139	0.237	0.148	0.239	0.153	0.246	0.195	0.278	0.165	0.252	0.165	0.252	0.201	0.317
	192	0.164	0.249	0.184	0.263	0.164	0.261	0.163	0.253	0.166	0.257	0.193	0.280	0.173	0.260	0.173	0.260	0.213	0.323
	336	0.180	0.267	0.194	0.276	0.178	0.272	0.181	0.274	0.185	0.275	0.206	0.296	0.188	0.276	0.188	0.276	0.259	0.358
	720	0.216	0.298	0.231	0.306	0.205	0.301	0.215	0.305	0.225	0.312	0.242	0.329	0.228	0.311	0.228	0.311	0.264	0.365
	Avg	0.178	0.263	0.198	0.276	0.172	0.268	0.177	0.268	0.182	0.272	0.209	0.296	0.188	0.275	0.188	0.275	0.234	0.341
Solar	96	0.212	0.221	0.250	0.252	0.208	0.246	0.210	0.210	0.216	0.308	0.291	0.379	0.216	0.261	0.246	0.345	0.874	0.709
	192	0.244	0.241	0.289	0.271	0.240	0.272	0.230	0.235	0.238	0.281	0.320	0.398	0.249	0.283	0.283	0.382	0.833	0.681
	336	0.273	0.255	0.317	0.286	0.262	0.290	0.259	0.255	0.256	0.289	0.353	0.416	0.268	0.295	0.281	0.373	0.930	0.718
	720	0.272	0.255	0.318	0.282	0.267	0.293	0.273	0.264	0.226	0.289	0.357	0.413	0.265	0.293	0.356	0.423	0.876	0.712
	Avg	0.250	0.243	0.294	0.273	0.244	0.275	0.243	0.241	0.234	0.292	0.330	0.402	0.250	0.283	0.292	0.381	0.878	0.705
Traffic	96	0.499	0.258	0.484	0.293	0.380	0.260	0.405	0.258	0.476	0.289	0.650	0.397	0.439	0.280	0.575	0.362	0.587	0.373
	192	0.523	0.261	0.478	0.287	0.394	0.267	0.430	0.266	0.465	0.290	0.599	0.371	0.450	0.282	0.613	0.386	0.586	0.362
	336	0.529	0.269	0.490	0.289	0.416	0.284	0.444	0.273	0.508	0.312	0.606	0.374	0.461	0.288	0.609	0.375	0.619	0.379
	720	0.543	0.292	0.526	0.303	0.457	0.300	0.480	0.293	0.565	0.312	0.646	0.395	0.498	0.310	0.644	0.398	0.662	0.397
	Avg	0.524	0.270	0.494	0.293	0.412	0.278	0.440	0.273	0.504	0.301	0.625	0.384	0.462	0.290	0.610	0.380	0.614	0.378
Weather	96	0.171	0.204	0.178	0.213	0.165	0.209	0.163	0.200	0.164	0.211	0.198	0.256	0.179	0.219	0.212	0.289	0.224	0.305
	192	0.215	0.246	0.222	0.251	0.215	0.255	0.211	0.246	0.210	0.254	0.238	0.295	0.225	0.259	0.279	0.344	0.330	0.388
	336	0.269	0.286	0.273	0.289	0.273	0.296	0.272	0.293	0.264	0.293	0.284	0.332	0.277	0.298	0.346	0.381	0.355	0.391
	720	0.345	0.338	0.343	0.336	0.353	0.349	0.345	0.342	0.343	0.345	0.348	0.385	0.351	0.346	0.398	0.411	0.460	0.454
	Avg	0.250	0.268	0.254	0.272	0.252	0.277	0.248	0.270	0.245	0.276	0.267	0.317	0.258	0.280	0.309	0.356	0.342	0.384
	1^{st} Count	5	31	4	6	10	1	7	8	11	0	0	0	7	0	3	0	0	0

Table 2. Statistics of the benchmark datasets. "Fea- tures" denotes the number of variables in each dataset, "Frequency" indicates the sampling in- terval, and "Seasonal" refers to the main period length of the data.

Datasets	Frequency	Features	Timesteps	Seasonal
ETTh1&2 [1]	1 h	7	17,420	24
ETTm1&2 [1]	5 min	7	69,680	96
Electricity [37]	1 h	321	26,304	24
Solar [38]	10 min	137	52,560	144
Traffic [23]	1 h	862	17,544	24
Weather [23]	10 min	21	52,696	144

- TimeMixer is a fully MLP-based model leveraging multiscale-mixing to disentangle seasonal and trend components, with PDM and FMM blocks for past extraction and future prediction.
- DLinear is a simple one-layer linear model with decomposition for time series forecasting.
- PatchTST is a Transformer-based model that segments sequences into channel-independent subseries-level patches to preserve local information and capture long-range dependencies efficiently.
- FEDformer is a frequency-enhanced Transformer that exploits sparse representations in Fourier basis to improve performance.
- Autoformer incorporates auto-correlation into a decomposition-based Transformer for improved sequence modeling.

4.4. Main Results

As shown in Table 1, our model achieves the best performance in terms of MAE on most datasets, but performs slightly worse on MSE. We attribute this to the Sigmoid Loss design, which makes the model more inclined to optimize MAE. In future work, incorporating MSE into the training objective may help alleviate this issue.

4.5. Ablation Study

4.5.1. Ablation Study on Model Architecture

As illustrated in Fig. 3, we investigate the impact of different architectural designs on experimental results. Four variants are considered:

- **PaDuM:** Seasonality → CNN stream, Trend → Mamba stream.
- **Reversed:** Seasonality → Mamba stream, Trend → CNN stream.
- **CNN only:** Seasonality → CNN stream, Trend → CNN stream.
- **Mamba only:** Seasonality → Mamba stream, Trend → Mamba stream.

The results demonstrate that the original PaDuM configuration achieves the best performance, benefiting from the CNN's strength in capturing local and periodic patterns and Mamba's efficiency in modeling long-term dependencies.

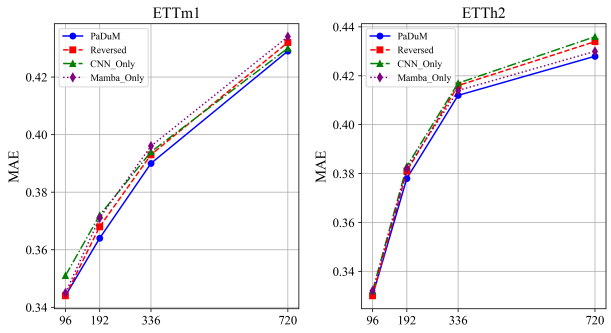

Fig. 3. Impact of different architectural designs. Look-back window $L = 96$ and prediction horizon $T = \{96, 192, 336, 720\}$.

4.5.2. *Ablation Study on type of Convolution*

In the CNN stream, we experimented with different types of convolution operations, including one-dimensional convolution (Conv1D) and twodimensional convolution (Conv2D). As shown in Table 3, we conducted ablation experiments on five datasets, and the results indicate that Conv2D outperforms Conv1D in most cases. This may be because Conv2D can better capture the interactions between time and variables, thereby improving the model's predictive capability.

4.5.3. *Ablation Study on Sigmoid Loss*

As shown in Fig. 4, the proposed Sigmoid Loss enhances forecasting performance, with notable improvements in long-term accuracy. These results confirm its effectiveness in guiding the model to focus on recent predictions. Overall, the loss function provides a stable optimization strategy for long-horizon forecasting.

In Eq. (15), three hyperparameters are involved: the decay rate k, the inflection point c, and the minimum weight l.

Table 3. Performance comparison between Conv2D and Conv1D on various datasets. Look-back window $L = 96$. The best performance is highlighted in red, and the second best is marked in lightblue with underlining.

Setting		Conv2D		Conv1D	
Metric		MSE	MAE	MSE	MAE
ETTh1	96	0.374	0.389	0.377	0.392
	192	0.434	0.420	0.436	0.424
	336	0.472	0.436	0.473	0.439
	720	0.514	0.473	0.512	0.474
	Avg	0.448	0.430	0.450	0.432
ETTh2	96	0.287	0.330	0.290	0.333
	192	0.360	0.378	0.363	0.382
	336	0.401	0.412	0.412	0.418
	720	0.408	0.428	0.423	0.437
	Avg	0.364	0.387	0.372	0.392
ETTm1	96	0.320	0.344	0.326	0.352
	192	0.372	0.374	0.373	0.373
	336	0.396	0.395	0.402	0.395
	720	0.459	0.429	0.467	0.431
	Avg	0.387	0.386	0.392	0.388
ETTm2	96	0.174	0.251	0.174	0.252
	192	0.240	0.296	0.239	0.296
	336	0.301	0.336	0.302	0.336
	720	0.396	0.392	0.399	0.393
	Avg	0.278	0.319	0.278	0.319
Weather	96	0.171	0.204	0.173	0.207
	192	0.215	0.246	0.220	0.248
	336	0.269	0.286	0.273	0.288
	720	0.345	0.338	0.343	0.336
	Avg	0.250	0.268	0.252	0.270

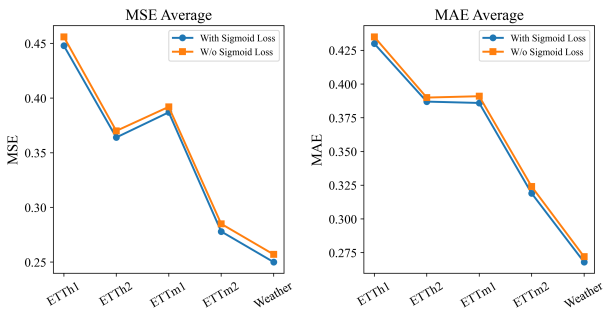

Fig. 4. Ablation study on sigmoid loss. Look-back window $L = 96$.

To investigate their effects on model performance, we conducted ablation experiments on five datasets; the results are reported in Table 4. We tested $k \in \{0.05, 2\}$, $c \in \{10, 20, 30\}$, and $l \in \{0.2, 0.5\}$. The results show only minor performance variations across combinations, indicating that the Sigmoid Loss is robust to these hyperparameters. Notably, the combination $k = 0.5, c = 30, l = 0.2$ performed well on most datasets; therefore, we fixed k and l and further evaluated the effect of c by extending its range to $\{40, 50, 60\}$.

As shown in Table Table 5, the model performance gradually decreases as c increases. This may be because a larger c assigns more weight to long-term predictions, causing the Sigmoid Loss to gradually degenerate into a standard MAE loss. To visualize the effect, we plot the Sigmoid Loss curves under different hyperparameter settings (see Fig. 5).

4.5.4. Ablation Study on Ability to Preserve Temporal Or- der

As shown in Table 6, to further investigate whether our model relies on the sequential order of the input series for forecasting, we designed two perturbation strategies. The first strategy, Half, swaps the first half and the second half of the original input sequence. The second strategy, Rand., randomly shuffles the entire input sequence. We conducted comparative experiments on three representative models that perform well under the original input setting.

On the ETTh1 dataset, which exhibits strong periodicity, all four models show relatively small and comparable performance degradation under the Half setting. However, under the Rand. setting, where the temporal dependency structure of the input sequence is severely disrupted, all models suffer significant performance deterioration, with xPatch and S-Mamba being particularly affected. The former is based on CNN and MLP architectures, while the latter is based on the Mamba architecture, both of which may rely more heavily on sequential order. In contrast, the Transformer-based CARD model benefits from the self-attention mechanism, making it less sensitive to input order, and thus its performance drop is smaller. Interestingly, our proposed hybrid CNN-Mamba model, which does not incorporate selfattention mechanisms, shows performance degradation comparable to CARD. This finding suggests that in time series forecasting tasks, the ability to effectively model temporal order may be more critical than reliance on specific architectural mechanisms, as it indicates that the model can capture the dynamic evolution of time series rather than simply memorizing static patterns.

On the Weather dataset, where periodicity is less prominent, the results present a different phenomenon. Under the Half setting, all four models experience substantial performance degradation, whereas under the Rand. setting, the performance drop is relatively smaller. This counterintuitive result highlights the inherent complexity of neural networks in temporal modeling and reveals the challenges in interpreting such models. At the same time, it provides new insights and directions for advancing the interpretability of time series forecasting models.

4.5.5. Ablation Study on Look-back Window

The look-back window length often has a significant impact on the performance of time series forecasting models. To investigate this effect, we conduct experiments on five benchmark datasets with lookback windows of 96, 192, 336, and 720 . The detailed results are reported in Table 7, and for better visualization, Table 6 plots the average MSE values across different settings. The results show that, in most datasets, increasing the look-back window generally improves model performance, with the best results achieved at a window length of 336 . However, further increasing the window to 720 does not yield additional improvements, suggesting that excessively long look-back windows may introduce redundant information, thereby limiting the model's learning effectiveness.

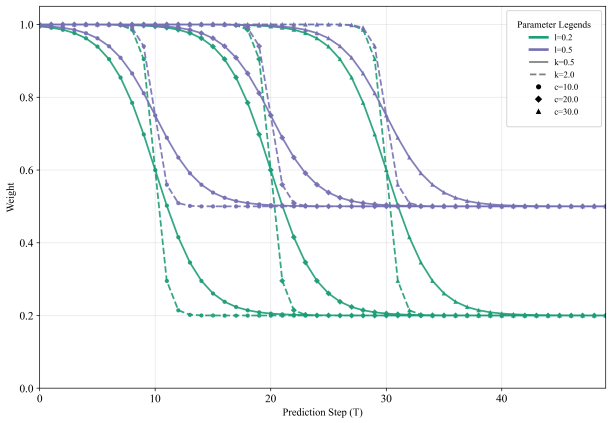

Fig. 5. Sigmoid Loss curves under different hyperparameter settings, illustrating the effect of k, c, and l on the weighting scheme. As c increases, the curve shifts to assign more weight to long-term predictions, which can reduce the model's focus on near-term predictions.

4.6. Evaluation of Model Efficiency and Effective- ness

We evaluate the training efficiency and effectiveness of different models on the ETTm2 dataset. To ensure fair comparison, all experimental settings follow the original implementations. As shown in Fig. 7, PaDuM achieves the best performance in terms of MAE while maintaining lower parameter count and shorter training time compared to other models. This demonstrates that PaDuM can achieve superior forecasting accuracy while preserving high training efficiency.

Fig. 8 reports the inference efficiency of our model under different look-back window lengths L. As shown, the one-pass forward latency remains stable at around 15 ms per sample, indicating that the proposed architecture is insensitive to the sequence length in terms of runtime. In contrast, the peak GPU memory consumption grows approximately linearly with L, which is expected due to the increased storage of intermediate states. These results demonstrate

Table 4. Ablation study of Sigmoid Loss hyperparameters k, c, and l on five datasets. The results show that PaDuM is robust to different hyperparameter settings, with only minor performance variations. Look-back window $L = 96$.The best performance is highlighted in red.

Setting		0.5,10,0.2		0.5,10,0.5		0.5,20,0.2		0.5,20,0.5		0.5,30,0.2		0.5,30,0.5		2,10,0.2		2,10,0.5		2,20,0.2		2,20,0.5		2,30,0.2		2,30,0.5	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.374	0.389	0.373	0.388	0.374	0.390	0.373	0.389	0.375	0.390	0.373	0.389	0.374	0.389	0.373	0.389	0.375	0.390	0.373	0.389	0.375	0.390	0.374	0.389
	192	0.434	0.420	0.433	0.420	0.434	0.419	0.433	0.419	0.434	0.419	0.433	0.419	0.434	0.420	0.433	0.420	0.434	0.419	0.433	0.419	0.434	0.419	0.433	0.419
	336	0.472	0.436	0.472	0.436	0.472	0.436	0.472	0.436	0.471	0.435	0.471	0.435	0.472	0.436	0.472	0.436	0.472	0.436	0.472	0.436	0.471	0.435	0.471	0.435
	720	0.513	0.473	0.506	0.469	0.478	0.458	0.505	0.469	0.509	0.471	0.517	0.474	0.527	0.480	0.540	0.483	0.523	0.477	0.533	0.481	0.513	0.474	0.530	0.480
	Avg	0.448	0.430	0.446	0.428	0.440	0.426	0.446	0.428	0.447	0.429	0.448	0.429	0.452	0.431	0.454	0.432	0.451	0.430	0.453	0.431	0.448	0.430	0.452	0.431
ETTh2	96	0.287	0.330	0.286	0.330	0.286	0.330	0.286	0.330	0.286	0.330	0.286	0.330	0.286	0.330	0.286	0.330	0.286	0.330	0.286	0.330	0.286	0.330	0.286	0.330
	192	0.360	0.378	0.360	0.378	0.361	0.379	0.360	0.378	0.362	0.379	0.361	0.378	0.360	0.378	0.360	0.378	0.361	0.379	0.361	0.378	0.362	0.379	0.361	0.378
	336	0.401	0.412	0.402	0.412	0.402	0.412	0.402	0.412	0.402	0.412	0.402	0.412	0.402	0.412	0.402	0.412	0.402	0.412	0.402	0.412	0.402	0.412	0.402	0.412
	720	0.408	0.428	0.412	0.430	0.411	0.430	0.411	0.430	0.409	0.429	0.411	0.430	0.421	0.434	0.424	0.436	0.419	0.434	0.423	0.435	0.419	0.434	0.423	0.435
	Avg	0.364	0.387	0.365	0.387	0.365	0.388	0.365	0.387	0.365	0.388	0.365	0.387	0.367	0.388	0.368	0.389	0.367	0.389	0.368	0.389	0.367	0.389	0.368	0.389
ETTm1	96	0.320	0.344	0.319	0.344	0.320	0.344	0.320	0.344	0.320	0.344	0.323	0.345	0.320	0.345	0.319	0.344	0.320	0.344	0.320	0.344	0.320	0.344	0.322	0.345
	192	0.368	0.371	0.368	0.371	0.368	0.371	0.369	0.372	0.367	0.370	0.369	0.371	0.367	0.371	0.368	0.371	0.367	0.371	0.369	0.372	0.367	0.370	0.369	0.371
	336	0.401	0.395	0.398	0.394	0.402	0.395	0.397	0.394	0.403	0.395	0.397	0.394	0.401	0.395	0.398	0.394	0.402	0.395	0.397	0.394	0.403	0.397	0.395	0.394
	720	0.459	0.429	0.462	0.430	0.460	0.429	0.462	0.430	0.459	0.429	0.460	0.432	0.460	0.430	0.460	0.431	0.460	0.430	0.460	0.430	0.460	0.431	0.460	0.431
	Avg	0.387	0.385	0.387	0.385	0.387	0.385	0.387	0.385	0.387	0.384	0.387	0.385	0.387	0.385	0.386	0.385	0.387	0.385	0.386	0.385	0.388	0.385	0.387	0.385
ETTm2	96	0.174	0.251	0.174	0.251	0.175	0.252	0.174	0.251	0.175	0.252	0.174	0.251	0.174	0.251	0.174	0.251	0.175	0.252	0.174	0.251	0.175	0.252	0.174	0.251
	192	0.240	0.296	0.238	0.294	0.239	0.295	0.240	0.296	0.239	0.295	0.240	0.296	0.240	0.296	0.238	0.294	0.239	0.295	0.240	0.296	0.239	0.295	0.240	0.296
	336	0.301	0.336	0.301	0.336	0.301	0.336	0.301	0.336	0.301	0.335	0.301	0.335	0.301	0.336	0.301	0.336	0.301	0.336	0.301	0.336	0.301	0.335	0.301	0.335
	720	0.396	0.391	0.396	0.392	0.396	0.391	0.396	0.392	0.396	0.391	0.396	0.392	0.397	0.392	0.397	0.392	0.398	0.392	0.397	0.392	0.398	0.392	0.397	0.392
	Avg	0.278	0.318	0.277	0.318	0.278	0.318	0.278	0.319	0.278	0.318	0.278	0.318	0.278	0.319	0.277	0.318	0.278	0.319	0.278	0.319	0.278	0.318	0.278	0.318
Weather	96	0.168	0.201	0.168	0.203	0.168	0.202	0.169	0.203	0.168	0.202	0.168	0.202	0.168	0.202	0.168	0.202	0.167	0.201	0.167	0.202	0.167	0.202	0.168	0.202
	192	0.215	0.245	0.215	0.245	0.214	0.245	0.214	0.245	0.214	0.244	0.215	0.245	0.215	0.245	0.215	0.245	0.214	0.244	0.214	0.245	0.214	0.244	0.215	0.245
	336	0.269	0.286	0.269	0.286	0.270	0.286	0.269	0.286	0.270	0.286	0.269	0.286	0.269	0.286	0.269	0.286	0.271	0.287	0.269	0.286	0.271	0.287	0.270	0.286
	720	0.345	0.339	0.345	0.339	0.345	0.339	0.345	0.339	0.343	0.337	0.346	0.340	0.346	0.339	0.345	0.339	0.345	0.339	0.346	0.339	0.345	0.338	0.346	0.340
	Avg	0.249	0.268	0.249	0.268	0.249	0.268	0.249	0.268	0.249	0.267	0.250	0.268	0.250	0.268	0.249	0.268	0.249	0.268	0.249	0.268	0.249	0.268	0.250	0.268
1^{St} Count		10	12	12	11	7	10	11	9	9	15	9	11	6	5	12	9	6	6	11	8	7	9	6	10

Table 5. Effect of the inflection point c on model performance. Look-back window $L = 96$. The best performance is highlighted in red.

Setting		0.5,10,0.2		0.5,10,0.5		0.5,20,0.2		0.5,20,0.5		0.5,30,0.2		0.5,30,0.5	
	Metric	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	96	0.374	0.389	0.374	0.390	0.375	0.390	0.376	0.393	0.376	0.394	0.376	0.395
	192	0.434	0.420	0.434	0.419	0.434	0.419	0.430	0.421	0.433	0.422	0.433	0.422
	336	0.472	0.436	0.472	0.436	0.471	0.435	0.474	0.438	0.474	0.438	0.474	0.438
	720	0.513	0.473	0.478	0.458	0.509	0.471	0.506	0.471	0.511	0.474	0.505	0.471
	Avg	0.448	0.430	0.440	0.426	0.447	0.429	0.446	0.431	0.448	0.432	0.447	0.432
ETTh2	96	0.287	0.330	0.286	0.330	0.286	0.330	0.288	0.331	0.288	0.331	0.289	0.331
	192	0.360	0.378	0.361	0.379	0.362	0.379	0.371	0.383	0.371	0.383	0.371	0.382
	336	0.401	0.412	0.402	0.412	0.402	0.412	0.408	0.415	0.408	0.415	0.407	0.415
	720	0.408	0.428	0.411	0.430	0.409	0.429	0.420	0.434	0.420	0.435	0.419	0.434
	Avg	0.364	0.387	0.365	0.388	0.365	0.388	0.372	0.391	0.372	0.391	0.372	0.390
ETTm1	96	0.320	0.344	0.320	0.344	0.320	0.344	0.320	0.345	0.320	0.345	0.321	0.346
	192	0.368	0.371	0.368	0.371	0.367	0.370	0.367	0.370	0.367	0.369	0.366	0.368
	336	0.401	0.395	0.402	0.395	0.403	0.395	0.394	0.392	0.394	0.392	0.393	0.391
	720	0.459	0.429	0.460	0.429	0.459	0.429	0.460	0.430	0.459	0.430	0.459	0.430
	Avg	0.387	0.385	0.387	0.385	0.387	0.384	0.385	0.384	0.385	0.384	0.385	0.384
ETTm2	96	0.174	0.251	0.175	0.252	0.175	0.252	0.174	0.252	0.174	0.252	0.174	0.252
	192	0.240	0.296	0.239	0.295	0.239	0.295	0.239	0.296	0.239	0.296	0.240	0.296
	336	0.301	0.336	0.301	0.336	0.301	0.335	0.304	0.339	0.305	0.339	0.306	0.339
	720	0.396	0.391	0.396	0.391	0.396	0.391	0.398	0.392	0.398	0.392	0.397	0.392
	Avg	0.278	0.318	0.278	0.318	0.278	0.318	0.279	0.320	0.279	0.320	0.279	0.320
Weather	96	0.168	0.201	0.168	0.202	0.168	0.202	0.170	0.203	0.170	0.203	0.169	0.202
	192	0.215	0.245	0.214	0.245	0.214	0.244	0.214	0.245	0.215	0.245	0.215	0.245
	336	0.269	0.286	0.270	0.286	0.270	0.286	0.269	0.285	0.269	0.285	0.269	0.285
	720	0.345	0.339	0.345	0.339	0.343	0.337	0.345	0.338	0.344	0.337	0.345	0.337
	Avg	0.249	0.268	0.249	0.268	0.249	0.267	0.250	0.268	0.250	0.268	0.250	0.267
	1^{st} Count	14	12	12	10	12	14	7	2	6	3	6	6

Table 6. Performance comparison of different models under shuffled input sequences. The look-back window is set to $L = 96$. *Ori.* denotes the original sequence, *Half* indicates swapping the first and second halves of the sequence, and Rand. represents random shuffling of the entire sequence. *Avg Drop* reports the average performance drop across four prediction lengths.

Method		PaDuM			xPatch			S-Mamba			CARD		
		Ori.	Half	Rand.	Ori.	Half	Rand.	Ori.	Half	Rand.	Ori.	Half	Rand.
	Metric	MSE MAE	MSE MAE	MSE MAE	MSE MAE	MSE MAE	MSE MAE	MSE MAE	MSE MAE	MSE MAE	MSE MAE	MSE MAE	MSE MAE
ETTh1	96	0.3770 .394	0.4230 .420	0.7900 .591	0.3730 .390	0.4250 .418	0.8540 .607	0.3880 .406	0.4240 .428	0.9080 .633	0.3800 .391	0.4130 .413	0.7380 .570
	192	0.4320 .422	0.4640 .439	0.8050 .610	0.4350 .422	0.4700 .442	0.8730 .622	0.4450 .441	0.4670 .456	0.9220 .647	0.4350 .420	0.4560 .434	0.7440 .575
	336	0.4730 .438	0.4940 .449	0.8060 .623	0.4840 .442	0.5050 .455	0.8720 .635	0.4900 .465	0.5050 .476	0.9150 .652	0.4790 .441	0.4910 .449	0.7580 .593
	720	0.5270 .480	0.5370 .486	0.7970 .625	0.4830 .464	0.4950 .470	0.9610 .689	0.5060 .497	0.5150 .503	0.9320 .672	0.4760 .462	0.4820 .466	0.7730 .616
	Avg	0.4520 .434	0.4800 .448	0.8000 .612	0.4440 .430	0.4740 .446	0.8900 .638	0.4570 .452	0.4780 .466	0.9190 .651	0.4420 .428	0.4600 .440	0.7530 .588
	Avg Drop	- -	6.193 .23	76.9941 .01	- -	6.763 .72	100.4548 .37	- -	4.603 .10	101.0944 .03	- -	4.072 .80	70.3637 .38
Weather	96	0.1710 .204	0.3040 .317	0.2170 .263	0.1780 .213	0.2860 .312	0.2170 .264	0.1650 .209	0.3010 .325	0.2210 .273	0.1630 .200	0.2890 .316	0.2280 .277
	192	0.2150 .246	0.3260 .339	0.2620 .297	0.2220 .251	0.3260 .342	0.2660 .299	0.2150 .255	0.3310 .347	0.2680 .307	0.2110 .246	0.3390 .351	0.2710 .309
	336	0.2690 .286	0.3690 .365	0.3090 .326	0.2730 .289	0.3650 .366	0.3100 .328	0.2730 .296	0.3760 .373	0.3160 .338	0.2720 .293	0.3820 .377	0.3190 .340
	720	0.3450 .338	0.4270 .401	0.3720 .366	0.3430 .336	0.4250 .404	0.3730 .367	0.3530 .433	0.4330 .408	0.3820 .377	0.3450 .342	0.4370 .412	0.3780 .376
	Avg	0.2500 .268	0.3560 .356	0.2900 .313	0.2540 .272	0.3500 .356	0.2920 .314	0.2520 .277	0.3600 .363	0.2970 .324	0.2480 .270	0.3620 .364	0.2990 .326
	Avg Drop	- -	42.4032 .84	16.0016 .79	- -	37.8030 .88	14.9615 .44	- -	42.8631 .05	17.8616 .97	- -	45.9734 .81	20.5620 .74

Table 7. Performance under different look-back window lengths $L \in \{96, 192, 336, 720\}$ across five datasets.

L		96		192		336		720	
Metric		MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
	96	0.377	0.394	0.372	0.393	0.366	0.395	0.377	0.407
	192	0.432	0.422	0.427	0.422	0.411	0.418	0.425	0.438
ETTh1	336	0.473	0.438	0.453	0.431	0.431	0.429	0.424	0.435
	720	0.528	0.480	0.579	0.503	0.501	0.489	0.552	0.512
	Avg	0.452	0.434	0.458	0.437	0.427	0.433	0.444	0.448
	96	0.287	0.330	0.300	0.342	0.310	0.349	0.288	0.339
	192	0.361	0.381	0.367	0.385	0.358	0.381	0.354	0.382
ETTh2	336	0.407	0.415	0.393	0.407	0.411	0.415	0.390	0.408
	720	0.421	0.434	0.410	0.427	0.397	0.423	0.514	0.511
	Avg	0.369	0.390	0.368	0.390	0.369	0.392	0.386	0.410
	96	0.317	0.344	0.308	0.337	0.299	0.336	0.291	0.338
	192	0.370	0.371	0.345	0.362	0.334	0.359	0.333	0.362
ETTm1	336	0.396	0.394	0.366	0.381	0.363	0.381	0.364	0.385
	720	0.460	0.430	0.447	0.423	0.429	0.417	0.424	0.414
	Avg	0.386	0.385	0.367	0.376	0.356	0.373	0.353	0.375
	96	0.174	0.252	0.173	0.250	0.166	0.248	0.166	0.252
	192	0.239	0.296	0.230	0.288	0.224	0.287	0.226	0.292
ETTm2	336	0.303	0.337	0.287	0.329	0.279	0.324	0.291	0.335
	720	0.397	0.392	0.367	0.379	0.356	0.374	0.367	0.384
	Avg	0.278	0.319	0.264	0.312	0.256	0.308	0.262	0.316
	96	0.169	0.203	0.159	0.192	0.151	0.186	0.145	0.184
	192	0.215	0.245	0.202	0.234	0.195	0.230	0.191	0.229
Weather	336	0.270	0.286	0.255	0.275	0.247	0.272	0.242	0.271
	720	0.345	0.338	0.331	0.329	0.319	0.324	0.323	0.329
	Avg	0.250	0.268	0.237	0.258	0.228	0.253	0.225	0.253

that our model achieves favorable scalability, especially for long-sequence forecasting scenarios.

4.6.1. Visualization of forecast results

We present the visualization of predictions from nine models on the ETTm2 dataset in Fig. 9. The experiments were conducted with a look-back window of $L = 96$ and a forecasting horizon of $T = 96$. It can be observed that, compared to other models, PaDuM better captures both the overall trend and local details of the time series. In contrast, other models exhibit larger deviations at certain time points, especially in regions with high volatility. The dual-stream architecture of PaDuM effectively combines local feature extraction with long-range dependency modeling, enabling superior performance on complex time series forecasting tasks.

5. Conclusions

In this paper, we propose PaDuM, a patch-based dual-stream network for efficient and accurate multivariate time series forecasting. By incorporating EMA-based decomposition, PaDuM effectively separates trend and seasonality components, thereby enhancing the discriminative power of input representations. Building on this, the model adopts

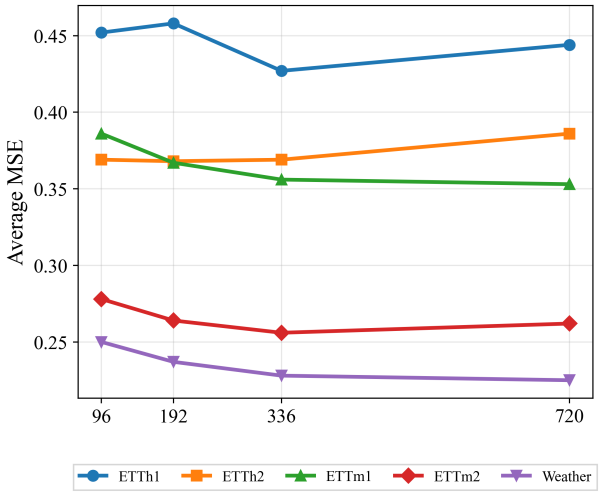

Fig. 6. Average MSE comparison under varying look-back window lengths. Results show that model performance improves with longer windows, peaking at 336, while further extension to 720 brings no additional benefit.

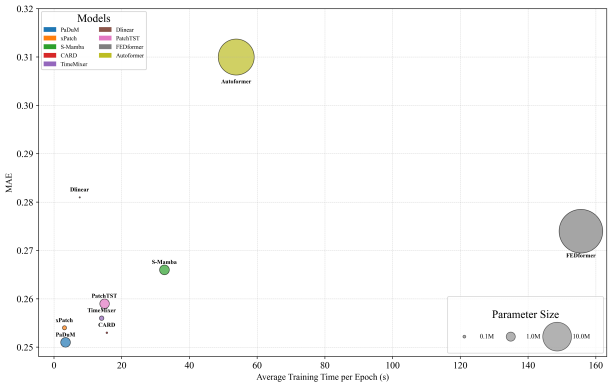

Fig. 7. Comparison of model training efficiency and effectiveness on the ETTm2 dataset. All settings follow the original implementations for fair comparison. PaDuM achieves the best MAE with fewer parameters and shorter training time, demonstrating its superior forecasting performance and high training efficiency.Look-back window $L = 96$ and prediction horizon $T = 96$.

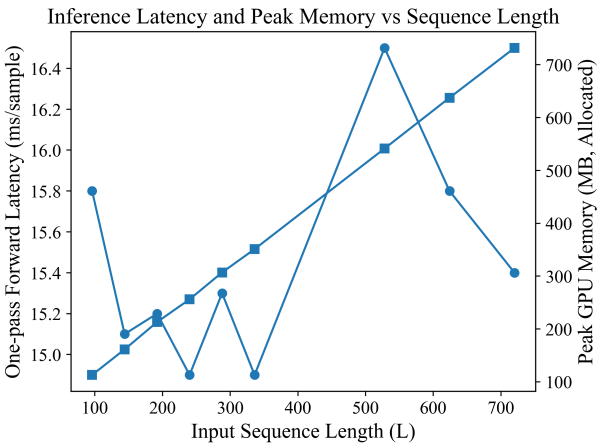

Fig. 8. Inference latency and peak GPU memory consumption under different input sequence lengths L. The latency remains nearly constant, while the memory usage grows approximately linearly with L.

a dual-stream architecture: CNN is employed to capture local fine-grained features within patches, while Mamba is used to model long-range dependencies across patches, fully leveraging the strengths of both architectures. Furthermore, a specially designed Sigmoid-normalized loss function is introduced to improve training stability. Extensive experiments on eight public datasets from domains such as electricity, transportation, and meteorology demonstrate that PaDuM achieves superior performance in terms of both prediction accuracy and robustness compared to existing state-of-the-art methods.

While the proposed PaDuM framework demonstrates strong performance in time series forecasting, there remain several promising directions for future research. First, the current Sigmoid Loss design places greater emphasis on optimizing MAE, which may lead to slightly inferior performance in terms of MSE. Future work could investigate hybrid or multiobjective loss formulations that jointly optimize MAE and MSE, aiming to achieve a better balance between robustness and overall forecasting accuracy.

Second, although PaDuM adopts a simple EMAbased decomposition for trend and seasonal decoupling due to its stability and effectiveness, more sophisticated data-driven decomposition strate-gies-such as STL decomposition, wavelet transforms, or Hodrick-Prescott filtering with learnable parameters-could be explored. Developing stable and learnable decomposition mechanisms that can be jointly optimized with the forecasting model remains an interesting yet challenging direction for future work.

Finally, the experiments reveal counterintuitive behaviors on datasets with weak or less prominent periodicity, highlighting the inherent complexity of neural networks in temporal modeling. Future studies may focus on improving the interpretability of PaDuM by incorporating explainable modules or visualizing learned temporal representations. Such efforts could provide deeper insights into the model's decision-making process and facilitate a better understanding of its behavior under diverse data characteristics.

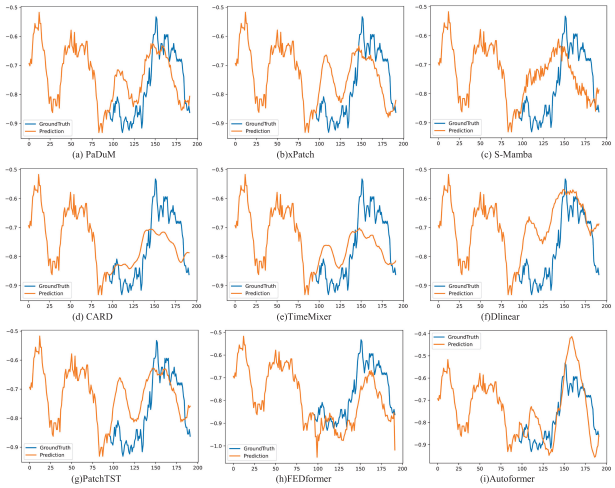

Fig. 9. Visualization of predictions from nine models on the ETTm2 dataset with a look-back window of $L = 96$ and prediction length $T = 96$. PaDuM captures both the overall trend and local details more accurately than other models, while other baselines show larger deviations in regions with high volatility.

Acknowledgment

This work was supported by the National Natural Science Foundation of China (U22A2026,62072097) / QIANKEHE PLATFORM TALENT BQW[2024]015/GZNU[2024]01.

References

[1] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, and W. Zhang. "Informer: Beyond efficient transformer for long sequence time-series forecasting". In: *Proceedings of the AAAI conference on artificial intelligence*. ***35***. 2021, 11106–11115. DOI: 10.1609/aaai.v35i12.17325.

[2] Z. Karevan and J. A. Suykens, (2020) *"Transductive LSTM for time-series prediction: An application to weather forecasting"* **Neural Networks *125***: 1–9. DOI: 10.1016/j.neunet.2019.12.030.

[3] L. Martín, L. F. Zarzalejo, J. Polo, A. Navarro, R. Marchante, and M. Cony, (2010) *"Prediction of global solar irradiance based on time series analysis: Application to solar thermal power plants energy production planning"* **Solar energy *84***(10): 1772–1781. DOI: 10.1016/j.solener.2010.07.002.

[4] X. Yin, G. Wu, J. Wei, Y. Shen, H. Qi, and B. Yin, (2021) *"Deep learning on traffic prediction: Methods, analysis, and future directions"* **IEEE Transactions on Intelligent Transportation Systems *23***(6): 4927–4943. DOI: 10.1109/TITS.2021.3054840.

[5] A. A. Laghari, V. V. Estrela, and S. Yin, (2024) *"How to collect and interpret medical pictures captured in highly challenging environments that range from nanoscale to hyperspectral imaging"* **Current Medical Imaging *20***(1): e281222212228. DOI: 10.2174/1573405619666221228094228.

[6] A. A. Laghari, Y. Sun, M. Alhussein, K. Aurangzeb, M. S. Anwar, and M. Rashid, (2023) *"Deep residual-dense network based on bidirectional recurrent neural network for atrial fibrillation detection"* **Scientific Reports *13***(1): 15109. DOI: 10.1038/s41598-023-40343-x.

[7] A. A. Laghari, S. Shahid, R. Yadav, S. Karim, A. Khan, H. Li, and Y. Shoulin, (2023) *"The state of art and review on video streaming"* **Journal of High Speed Networks *29***(3): 211–236. DOI: 10.3233/JHS-222087.

[8] A. A. Laghari, V. V. Estrela, H. Li, Y. Shoulin, A. A. Khan, M. S. Anwar, A. Wahab, and K. Bouraqia, (2024) *"Quality of experience assessment in virtual/augmented reality serious games for healthcare: A systematic literature review"* **Technology and Disability *36***(1-2): 17–28. DOI: 10.3233/TAD-230035.

[9] S. Yin, H. Li, L. Teng, A. A. Laghari, A. Almadhor, M. Gregus, and G. A. Sampedro, (2024) *"Brain CT image classification based on mask RCNN and attention mechanism"* **Scientific Reports *14***(1): 29300. DOI: 10.1038/s41598-024-78566-1.

[10] M. A. Munir, R. A. Shah, M. Ali, A. A. Laghari, A. Almadhor, and T. R. Gadekallu, (2024) *"Enhancing Gene Mutation Prediction with Sparse Regularized Autoencoders in Lung Cancer Radiomics Analysis"* **IEEE Access**: DOI: 10.1109/ACCESS.2024.3523330.

[11] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin. "Attention is all you need". In: *Advances in Neural Information Processing Systems*. ***30***. 2017.

[12] A. Gu, K. Goel, and C. Ré, (2021) *"Efficiently modeling long sequences with structured state spaces"* **arXiv preprint arXiv:2111.00396**: DOI: 10.48550/arXiv.2111.00396.

[13] J. T. Smith, A. Warrington, and S. W. Linderman, (2022) *"Simplified state space layers for sequence modeling"* **arXiv preprint arXiv:2208.04933**: DOI: 10.48550/arXiv.2208.04933.

[14] A. Gu, K. Goel, A. Gupta, and C. Ré, (2022) *"On the Parameterization and Initialization of Diagonal State Space Models"* **Advances in Neural Information Processing Systems *35***: 35971–35983.

[15] A. Gu, I. Johnson, K. Goel, K. Saab, T. Dao, A. Rudra, and C. Ré, (2021) *"Combining recurrent, convolutional, and continuous-time models with linear state space layers"* **Advances in neural information processing systems *34***: 572–585.

[16] A. Katharopoulos, A. Vyas, N. Pappas, and F. Fleuret. "Transformers are rnns: Fast autoregressive transformers with linear attention". In: *International conference on machine learning*. PMLR. 2020, 5156–5165.

[17] T. Dao, D. Y. Fu, K. K. Saab, A. W. Thomas, A. Rudra, and C. Ré. "Hungry Hungry Hippos: Towards Language Modeling with State Space Models". In: *International Conference on Learning Representations*. 2023. DOI: 10.48550/arXiv.2212.14052.

[18] J. T. H. Smith, A. Warrington, and S. W. Linderman. "Simplified State Space Layers for Sequence Modeling". In: *International Conference on Learning Representations*. 2023. DOI: 10.48550/arXiv.2208.04933.

[19] A. Gu and T. Dao, (2024) *"Mamba: Linear-time sequence modeling with selective state spaces"* **Conference on Language Modeling**:

[20] A. Dosovitskiy, L. Beyer, A. Kolesnikov, D. Weissenborn, X. Zhai, T. Unterthiner, M. Dehghani, M. Minderer, G. Heigold, S. Gelly, et al. "An Image is Worth 16x16 Words: Transformers for Image Recognition at Scale". In: *International Conference on Learning Representations*. 2020.

[21] E. S. Gardner Jr, (1985) *"Exponential smoothing: The state of the art"* **Journal of Forecasting *4***(1): 1–28. DOI: 10.1002/for.3980040103.

[22] J. Tao, L. Cao, H. Wang, C. Xie, J. Li, and L. Zhou, (2026) *"PaDuM: Patch-Based Dual-Stream Network with CNN and Mamba for Time Series Forecasting"* **Submitted to the 2026 IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP 2026)**: DOI: 10.48550/arXiv.2411.05793.

[23] H. Wu, J. Xu, J. Wang, and M. Long. "Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting". In: *Advances in Neural Information Processing Systems*. ***34***. 2021, 22419–22430.

[24] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, and R. Jin. "Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting". In: *International Conference on Machine Learning*. ***162***. Proceedings of Machine Learning Research. PMLR. 2022, 27268–27286.

[25] A. Zeng, M. Chen, L. Zhang, and Q. Xu. "Are transformers effective for time series forecasting?" In: *Proceedings of the AAAI Conference on Artificial Intelligence*. ***37***. 2023, 11121–11128. DOI: 10.1609/aaai.v37i9.26317.

[26] A. Stitsyuk and J. Choi. "xPatch: Dual-Stream Time Series Forecasting with Exponential Seasonal-Trend Decomposition". In: *Proceedings of the AAAI Conference on Artificial Intelligence*. ***39***. 2025, 20601–20609. DOI: 10.1609/aaai.v39i19.34270.

[27] Y. Nie, H. N. Nguyen, P. Sinthong, and J. Kalagnanam. "A Time Series is Worth 64 Words: Long-term Forecasting with Transformers". In: *International Conference on Learning Representations*. 2023. DOI: 10.48550/arXiv.2211.14730.

[28] Y. Zhang and J. Yan. "Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting". In: *The Eleventh International Conference on Learning Representations (ICLR)*. 2022.

[29] L. Sifre and S. Mallat, (2014) *"Rigid-motion scattering for texture classification"* **International Journal of Computer Vision**: DOI: 10.48550/arXiv.1403.1687.

[30] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015.

[31] S. Ioffe and C. Szegedy. "Batch normalization: Accelerating deep network training by reducing internal covariate shift". In: *International conference on machine learning*. PMLR. 2015, 448–456.

[32] A. G. Howard, M. Zhu, B. Chen, D. Kalenichenko, W. Wang, T. Weyand, M. Andreetto, and H. Adam, (2017) *"Mobilenets: Efficient convolutional neural networks for mobile vision applications"* **Computer Vision and Pattern Recognition**: DOI: 10.48550/arXiv.1704.04861.

[33] F. Chollet. "Xception: Deep learning with depthwise separable convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017.

[34] A. Trockman and J. Z. Kolter, (2022) *"Patches Are All You Need?"* **Transactions on Machine Learning Research**: DOI: 10.48550/arXiv.2201.09792.

[35] Z. Wang et al., (2025) *"Is mamba effective for time series forecasting?"* **Neurocomputing *619***: 129178. DOI: 10.1016/j.neucom.2024.129178.

[36] M. A. Ahamed and Q. Cheng, (2024) *"TimeMachine: A Time Series is Worth 4 Mambas for Long-term Forecasting"* **ECAI 2024: 27th European Conference on Artificial Intelligence**: DOI: 10.3233/faia240677.

[37] A. Trindade, (2015) *"ElectricityLoadDiagrams20112014"* **UCI Machine Learning Repository *10***: C58C86.

[38] G. Lai, W.-C. Chang, Y. Yang, and H. Liu. "Modeling long-and short-term temporal patterns with deep neural networks". In: *The 41st international ACM SIGIR conference on research & development in information retrieval*. 2018, 95–104. DOI: 10.1145/3209978.3210006.

[39] X. Wang, T. Zhou, Q. Wen, J. Gao, B. Ding, and R. Jin. "CARD: Channel Aligned Robust Blend Transformer for Time Series Forecasting". In: *The Twelfth International Conference on Learning Representations*. 2024. DOI: 10.48550/arXiv.2305.12095.

[40] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J. Y. Zhang, and J. ZHOU. "TimeMixer: Decomposable Multiscale Mixing for Time Series Forecasting". In: *International Conference on Learning Representations (ICLR)*. 2024. DOI: 10.48550/arXiv.2405.14616.

[41] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, and M. Long. "TimesNet: Temporal 2D-Variation Modeling for General Time Series Analysis". In: *International Conference on Learning Representations*. 2023. DOI: 10.48550/arXiv.2210.02186.