Knowledge-Based Systems 325 (2025) 113971

Contents lists available at ScienceDirect

Knowledge-Based Systems

journal homepage: www.elsevier.com/locate/knosys

mLANet: An efficient recurrent neural network for long-term time series forecasting

Jihua Jiang [a], Bin Jiang [a,*], Yong Wang [a], Duoqian Miao [b]

[a] School of Artificial Intelligence, Chongqing University of Technology, Chongqing, China
[b] School of Computer Science and Technology, Tongji University, Shanghai, China

ARTICLE INFO

Keywords:
Time series forecasting
Multivariate time series
Long-term dependencies

ABSTRACT

In recent years, Transformer-based deep learning models have attracted widespread attention in time series forecasting due to their remarkable ability to capture interactions within long sequences. However, their self-attention mechanism struggles to effectively represent temporal continuity and causal relationships. To address this issue, we propose an innovative model based on recurrent neural networks, called mLANet. This model is designed to better capture long-sequence dependencies while fully reflecting temporal continuity and causality. In mLANet, we developed the AECCM module, which transforms one-dimensional time series data into a two-dimensional format. By integrating convolutional neural networks with attention mechanisms, this module simultaneously extracts both local and global features. The output of the mLSTM is fused with the AECCM features through element-wise multiplication, enabling the model to dynamically emphasize critical information. This approach enhances the model's feature representation and improves its ability to capture long-term dependencies. Furthermore, we introduce a novel time series data transformation strategy that processes multiple time windows in parallel. This approach significantly reduces computational complexity while strengthening the model's ability to detect local patterns. Experimental results demonstrate that mLANet achieves state-of-the-art predictive performance across seven publicly available multivariate time series datasets. Its exceptional accuracy and robustness highlight its superiority in time series forecasting tasks. Code is available at: https://github.com/jiangjihua8/mLANet.

1. Introduction

The 21st century has ushered in the data era, characterized by vast quantities and diverse forms of data. This is particularly evident in the rapid advancements of Internet of Things (IoT) technology, where sensor devices have seen remarkable improvements in their numbers, types, and data-gathering capabilities. Therefore, in IoT systems with a large number of sensors, traditional statistical methods such as autoregression (AR) [1], moving average (MA) [2], autoregressive moving average (ARMA) [3], and autoregressive integrated moving average (ARIMA) [4] are no longer adequate to cope with such complexity. Consequently, an increasing number of data scientists are shifting their focus to research on time series forecasting based on deep learning. Moreover, deep learning-based time series forecasting models have been widely applied in numerous domains, such as transportation and logistics [5], energy management [6], finance [7], meteorology and environmental science [8], and healthcare [9].

In recent years, variant models based on the Transformer architecture (such as Informer [10], ETSformer [11], etc.) have dominated the field of long-term time series forecasting (LTSF). These models significantly enhance the ability to model long-range dependencies through self-attention mechanisms and parallelized computation. However, these models still face notable challenges in practical applications. First, the self-attention mechanism has limitations in capturing local time series patterns, and its computational complexity grows quadratically with sequence length, restricting its practicality in scenarios involving extremely long sequences [12]. Second, the Transformer [13] relies on positional encoding to introduce temporal information, which may lead to the implicit loss of dynamic time features due to encoding biases. Although existing studies have partially addressed these issues through sparse attention [10,14] or chunking strategies [15], the models' ability to capture the inherent causality and gradual evolution patterns of time series remains insufficient [12]. In contrast, recurrent neural networks (RNNs) [16] and their derivatives (such as LSTM [17] and GRU [18]) are naturally suited to model the causal dependencies and continuous state evolution of time series due to their sequential processing nature. However, traditional LSTM is limited by its scalar memory units and recursive computation mode,

* Corresponding author.
E-mail addresses: 52242313206@stu.cqut.edu.cn (J. Jiang), jb20200132@cqut.edu.cn (B. Jiang).

https://doi.org/10.1016/j.knosys.2025.113971
Received 1 April 2025; Received in revised form 24 May 2025; Accepted 13 June 2025
Available online 25 June 2025

making it difficult to effectively capture dependencies over very long spans. Moreover, in long-term time series forecasting, traditional models face several challenges. First, capturing long-term dependencies is difficult [19]; when dealing with dependencies spanning extended periods, traditional models may struggle due to gradient vanishing or exploding problems, hindering their ability to effectively capture long-term dependencies [20]. Second, there is an information flow bottleneck; in deep networks, information must pass through multiple layers, which can impede the flow and affect the model's capacity to handle long-sequence data. Additionally, insufficient feature fusion poses a challenge; multidimensional time series data requires effective mechanisms to integrate features and capture complex relationships between them. Finally, traditional models may encounter computational bottlenecks when processing long sequences due to recursive calculations, resulting in low training efficiency.

To address these issues, we propose the mLSTM with Attention-Enhanced Convolutional Fusion Network (mLAN) framework to tackle the difficulties faced by Transformer models in modeling the inherent causality and gradual evolution patterns of time series, as well as the challenges recurrent neural networks encounter in capturing long-term dependencies. mLAN stacks multiple mLSTM layers [21], where each layer utilizes matrix memory units to store larger-scale information, significantly enhancing the model's memory capacity and improving its ability to model dependencies over extended time spans. The cascaded mLSTM, through matrix memory and interlayer information transfer, enables the second mLSTM layer to process and optimize the information flow passed from the first mLSTM layer, effectively capturing long-term dependencies and addressing the computational bottlenecks of traditional LSTM in long sequences. We transform the time series data from (B, N, L) to (B, C, N, L//C) (where B represents batch size, C denotes the number of channels, N indicates the number of features, and L stands for sequence length). By processing multiple time windows in parallel, with each channel handling different segments of the time series, and integrating the Attention-Enhanced Convolutional Context Module (AECCM)—which combines convolutional layers and an attention mechanism—we fuse multidimensional features of the time series data. This fusion approach captures complex relationships between features and further enhances the model's ability to extract information and learn features. The model can comprehensively extract both local details and global trends, strengthening its capacity to capture dependencies in complex time series patterns. Through these methods, we effectively address multiple challenges in long-term time series forecasting, improving the model's predictive performance and computational efficiency.

The main contributions of this paper are summarized as follows:

- We propose the mLANet model framework, which effectively captures the continuity and causality of long-term multivariate time series. With strong temporal context modeling capabilities, it addresses the challenges traditional models face in modeling long-sequence dependencies more efficiently.
- We introduce a method to transform time series data from 1D to 2D by splitting the sequence length. This leverages the feature extraction power of convolutional neural networks to process multiple time windows in parallel, effectively capturing local patterns and short-term dependencies while reducing computational complexity and improving model performance.
- We design the AECCM module, which uses convolutional layers and an attention mechanism on 2D data to fuse multidimensional features of time series. This enhances the ability to extract information and learn features, enabling a more comprehensive extraction of local details and global trends, and strengthening the capacity to capture dependencies in complex time series patterns.
- Additionally, extensive experiments demonstrate that mLANet achieves state-of-the-art performance on seven publicly available multivariate datasets, while also exhibiting excellent accuracy and robustness.

2. Related work

Time series forecasting has undergone a rapid evolution over the past few decades, shifting from traditional statistical models to deep learning models, and has gradually become a core research focus in many fields. Early studies mainly relied on statistical methods, such as Autoregressive (AR) [1], Moving Average (MA) [2], and their combination, the Autoregressive Moving Average (ARMA) model [3]. These models provide good interpretability and predictive performance for univariate and stationary sequences, but they often require assumptions of linearity or stationarity in the data distribution. When handling multivariate long sequences or non-stationary time series, these methods face significant limitations. To tackle the challenges of non-stationarity and multivariate data, researchers introduced the Autoregressive Integrated Moving Average (ARIMA) model [4], which manages non-stationary sequences through differencing operations. However, these statistical approaches still struggle to capture nonlinear relationships and are sensitive to noise when dealing with complex long-sequence dependencies and multidimensional time features.

With the rise of deep learning, numerous studies have emerged using neural networks for time series forecasting, achieving impressive results. RNN-based models (such as SegRNN [22], and LSTNet [23]) excel at capturing feature relationships and can take advantage of the sequential nature of time series. However, when sequences become too long, these models encounter problems like gradient vanishing or difficulties with parallel computation, which reduce their efficiency or predictive performance in long-sequence modeling. To improve the ability to capture long-term dependencies, researchers introduced the attention mechanism and, inspired by the Transformer structure from natural language processing, developed methods for modeling long dependencies using multi-head self-attention (such as LogSparse Transformer [24], Informer [10], and Autoformer [25]). Transformers perform exceptionally well in parallel processing and capturing long-distance dependencies. However, their positional encoding struggles to express temporal continuity and causality, and when faced with real-world multivariate time series data, they often require additional modeling of complex relationships across multiple scales or variables. As a result, a series of improved models have been proposed (such as PatchTST [15], and iTransformer [26]). MLP-based models (like TSMixer [27] and DLinear [12]) and CNN-based models (like MICN [28], and TimesNet [29]) have also shown excellent performance in time series forecasting.

Currently, deep learning has made significant progress in computer vision and natural language processing, with most methods designed for 2D data. To apply these techniques to 1D sequence data, researchers have proposed innovative approaches, such as converting 1D sequences into 2D images [30,31], or explicitly transforming 1D time series into 2D tensors using multi-periodicity, as seen in TimesNet [29].

Building on these ideas, we aim to design a completely new network that can replace Transformers in addressing long-sequence dependencies while fully capturing the temporal continuity and causality of multivariate long sequences. Additionally, we introduce a novel method for converting 1D time series data into a 2D format for time series forecasting.

3. mLANet

Our overall network framework is shown in Fig. 1. It improves the PRfomer [33] network as the base network to create the mLANet architecture. The Pyramidal RNN Embedding module, which is derived from PRformer, serves as a feature extractor to effectively integrate multi-scale temporal dependency information. Our proposed mLAN architecture mainly consists of the mLSTM and AECCM blocks, designed to tackle the challenge of capturing long-term dependencies in multivariate time series forecasting. The two-layer mLSTM structure allows each layer to use matrix memory units to store larger-scale

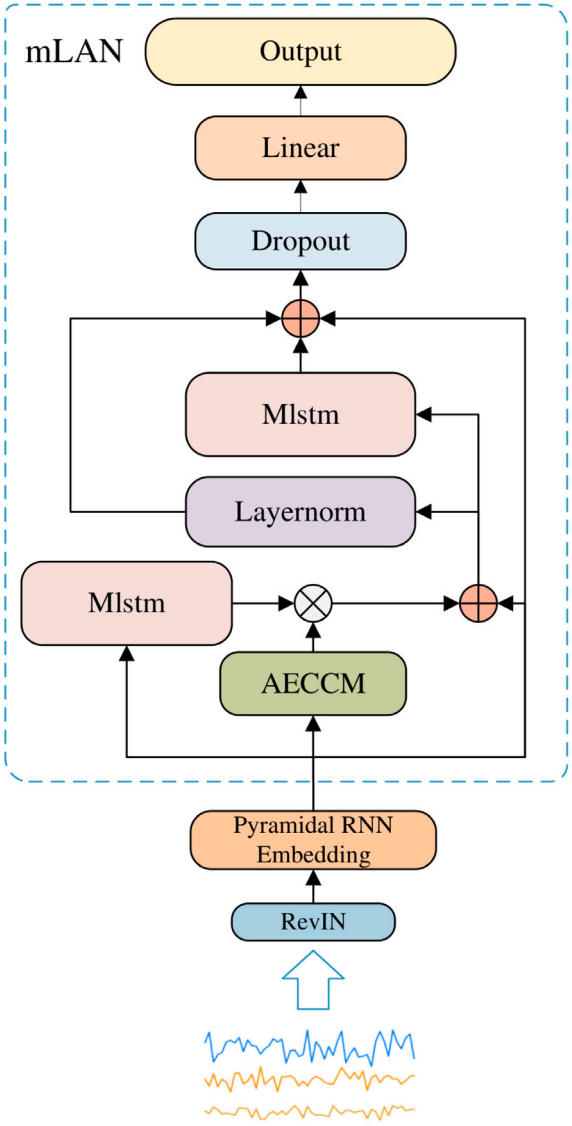

Fig. 1. The overall structure of the mLANet model.

information. This greatly enhances the model's memory capacity. It also improves the ability to model dependencies over long time spans. The two-layer mLSTM structure, through the matrix memory mechanism and inter-layer information transmission strategy, This significantly enhances the model's memory capacity and improves its ability to model dependencies over long time spans. The two-layer mLSTM structure achieves its functionality through a matrix memory mechanism and an inter-layer information transmission strategy. The first layer directly processes the output sequence from the Pyramidal RNN Embedding to extract basic sequence features, which can capture local temporal dependencies, and identify low-level patterns. In contrast, the second layer focuses on the abstracted features from the first layer to recognize more abstract patterns and model longer-range dependencies. This hierarchical processing approach enables the model to learn higher-level representations, effectively integrate information across longer sequence spans, and capture more complex temporal relationships. The inter-layer information transmission ensures that this information flows efficiently between layers and is further refined, thereby more effectively capturing long-term dependencies. The output of the first mLSTM layer is fused with the output of the AECCM module through element-wise multiplication to highlight important features, while enhancing the model's adaptability to sequence data.

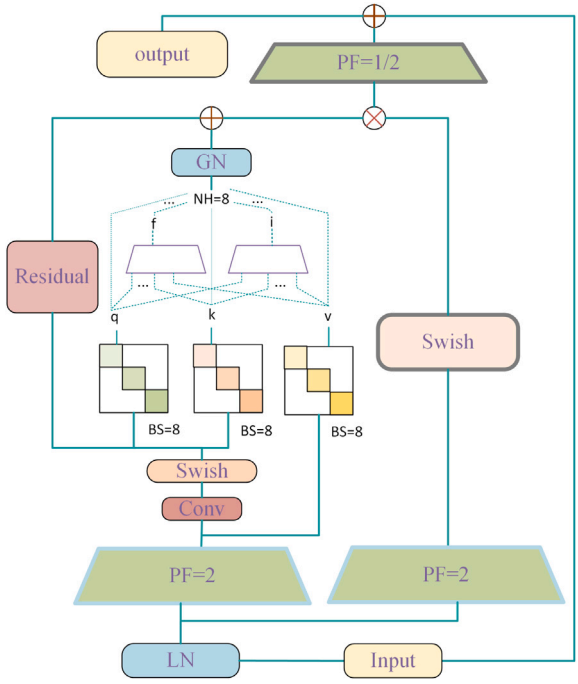

Fig. 2. Schematic of the mLSTM unit - pre-up-projection stage: The input is embedded within a pre-LayerNorm residual framework and first up-projected with a factor of 2, generating inputs for an external output gate and the mLSTM unit. The mLSTM unit's input undergoes dimension-wise causal convolution (kernel size 8) before entering a learnable residual connection. Inputs q and k are derived from block-diagonal projection matrices (block size 8), while the value v is directly passed, bypassing convolution. After mLSTM sequence mixing, the output is normalized using GroupNorm [32], applying head-wise layer normalization for each of the 8 heads. Finally, the learnable residual input is added back, the result is element-wise gated with the external output gate, and the output is down-projected.

3.1. Problem definition

In long-term time series forecasting tasks, based on the historical multivariate time series (MTS) $X = \{X_t\}_{t=1}^{l} \in \mathbb{R}^{l\times N}$, we predict the future time series $Y = \{Y_t\}_{t=l+1}^{l+h} \in \mathbb{R}^{h\times N}$. Here, l is the length of the historical look-back window, N is the feature dimension, and h is the length of the future prediction window. The goal of long-term time series tasks is to maximize the prediction horizon h.

3.2. Mlstm module

mLSTM (Matrix Long Short-Term Memory) originates from an enhanced variant within the xLSTM [21] (Extended Long Short-Term Memory) family, The overall structure of the mLSTM module as show in Fig. 2, designed to address the limitations of traditional LSTM (Long Short-Term Memory) networks in terms of storage capacity and parallelization capabilities. By upgrading the memory unit from a scalar to a matrix and incorporating a covariance update rule, it significantly boosts memory capacity and parallel processing efficiency, delivering superior performance and efficiency in handling complex sequential tasks such as language modeling. While the memory unit in traditional LSTM is a scalar $c \in \mathbb{R}$, mLSTM extends it to a matrix $M \in \mathbb{R}^{d\times d}$, greatly enhancing storage capacity to capture more intricate long-term dependencies. The memory matrix M_t is dynamically updated through the covariance update rule, based on the theoretical covariance update rule [34,35], expressed as:

$$M_t = M_{t-1} + v_t k_t^{\top} \tag{1}$$

Specifically, the covariance update rule dynamically adjusts and stores memory by continuously updating the memory matrix M_t based on the outer product of the new key–value pair v_t and k_t.

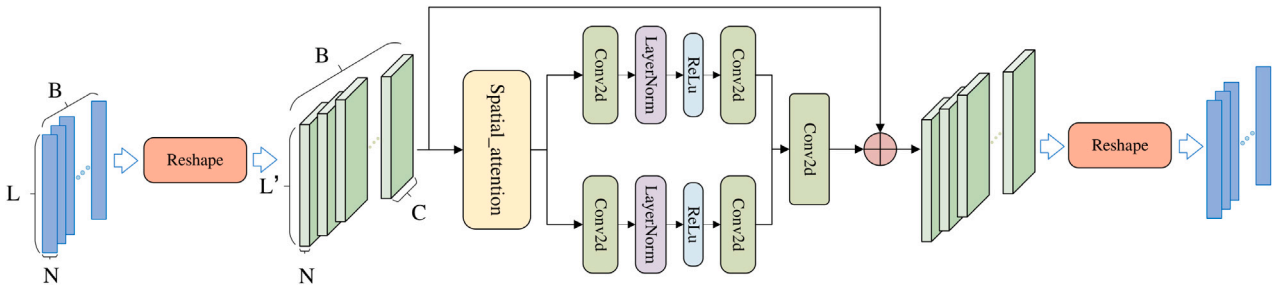

Fig. 3. The overall structure of the AECCM module.

The memory matrix M_t undergoes dynamic updating using a covariance rule, expressed as:

$$M_t = \phi_t M_{t-1} + \iota_t \upsilon_t \kappa_t^\top \tag{2}$$

where ϕ_t serves as the forget gate to regulate memory decay, ι_t acts as the input gate to control the intensity of new information incorporation, and υ_t and κ_t represent the value and key vectors, respectively. This mechanism bears similarities to fast weight programmers and draws inspiration from the attention mechanism in Transformers, enabling efficient storage and retrieval of key–value pair data. Furthermore, mLSTM eliminates the recurrent connections between hidden states (i.e., memory mixing) found in traditional LSTM, achieving full parallelization and substantially improving computational efficiency for long-sequence tasks. It retains and refines the gating mechanism, including the forget gate ϕ_t, input gate ι_t, and output gate ω_t (used to scale retrieved vectors), while introducing exponential gating and stabilization techniques—such as the normalization state ν_t and auxiliary state μ_t—to mitigate numerical instability issues, such as exponential overflow.

The core formulas of the mLSTM forward pass, detailing its computational process, are as follows:

Normalization State Update

$$\eta_t = \phi_t \eta_{t-1} + \iota_t \kappa_t \tag{3}$$

where η_t is the normalization state used to track the cumulative strength of gating for stabilizing computation, κ_t is the key vector.

Hidden State Computation

$$\zeta_t = \omega_t \odot \hat{\zeta}_t, \quad \hat{\zeta}_t = \frac{S_t \psi_t}{\max\{|\eta_t^\top \psi_t|, 1\}} \tag{4}$$

where ζ_t is the hidden state at time step t, ω_t is the output gate scaling the intermediate hidden state $\hat{\zeta}_t$ element-wise, $\hat{\zeta}_t$ is the intermediate hidden state retrieved from the memory matrix S_t using the query vector ψ_t, $\max\{|\eta_t^\top \psi_t|, 1\}$ is the normalization factor preventing division by near-zero values.

Computation of Inputs, Keys, Values, and Gates Retrieval vector:

$$\psi_t = W_q x_t + b_q \tag{5}$$

Key vector:

$$\kappa_t = \frac{1}{\sqrt{d}} W_k x_t + b_k \tag{6}$$

Value vector:

$$\nu_t = W_\upsilon x_t + b_\upsilon \tag{7}$$

Input gate:

$$\iota_t = \exp(\tilde{\iota}_t), \quad \tilde{\iota}_t = w_i^\top x_t + b_i \tag{8}$$

Forget gate:

$$\phi_t = \sigma(\tilde{\phi}_t) \text{ or } \exp(\tilde{\phi}_t), \quad \tilde{\phi}_t = w_f^\top x_t + b_f \tag{9}$$

Output gate:

$$\omega_t = \sigma(\tilde{\omega}_t), \quad \tilde{\omega}_t = W_o x_t + b_o \tag{10}$$

where x_t is the input at time step t, W_q, W_k, W_v, W_o are weight matrices, $b_q, b_k, b_v, b_i, b_f, b_o$ are bias terms, σ is the sigmoid activation function, exp is the exponential function used for exponential gating, and d represents the dimension of the key vector k_t, that is, $k_t \in \mathbb{R}^d$.

3.3. Attention-enhanced convolutional context module

Our proposed Attention-Enhanced Convolutional Context Module (AECCM) consists of four main components, as shown in Fig. 3 : (1) Reshape layer, (2) Spatial attention layer, (3) Dual convolutional sequence layers, and (4) Residual connection.

Reshape layer: The reshape layer transforms the input data from (B, N, L) into $(B, C, N, L/C)$, converting time series data from a 1D representation to a 2D one. By decomposing the sequence length into multiple channels, the model splits the original sequence into several subsequences, processing them in parallel across different channels. This decomposition boosts computational efficiency and enables the model to learn complementary feature representations across channels. In the transformed 2D time series data, the Convolutional Neural Network (CNN) fully leverages its strong feature extraction capabilities. Convolutional kernels slide across both the 'features' and 'time steps' dimensions to capture local patterns. This local receptive field allows the model to automatically learn short-term dependencies and feature interactions within the time series, while also capturing complex interactions across channels and time steps. Such multi-dimensional feature extraction reveals hidden patterns in the time series, such as periodicity, trend changes, or feature correlations. Additionally, it enables the model to learn joint behaviors of different features across various time periods, enhancing its ability to model complex data. Here, B represents the batch size, N denotes the number of features, L indicates the sequence length, and C signifies the number of channels.

Spatial attention layer: The spatial attention mechanism, implemented via 2D convolution, generates global contextual information within the 2D space of 'features' and 'time steps.' This global context offers the model a 'bird's-eye view,' enabling it to grasp the overall characteristics of the sequence beyond just local patterns.

Dual convolutional sequence layers: as illustrated in Fig. 3. In each convolutional sequence layer, the first convolutional layer extracts features by reducing the number of channels, followed by layer normalization and a nonlinear activation function. Subsequently, a second convolutional layer restores the number of channels to the original dimension. Specifically, the extracted global contextual information is first processed separately by a 3×3 convolution module and a 1×1 convolution module. For 2D data, the 3×3 convolution kernel captures local dependencies between time and features, extracting short-term patterns and identifying correlations between features in adjacent time steps. Meanwhile, the 1×1 convolution module performs feature fusion across channels, learns relationships between different channels, enhances feature abstraction, and improves the model's understanding

Table 1
Details of datasets used for experimentation.

Datasets	Frequency	Features	Timesteps
ETTh1&ETTh2	1 h	7	17,420
ETTm1&ETTm2	5 min	7	69,680
Weather	10 min	21	52,696
Electricity	1 h	321	26,304
Traffic	1 h	862	17,544

of multi-level patterns in time series. Following this, a 2D 1×1 convolutional layer fuses the outputs from the 3×3 and 1×1 convolution modules to generate the final output features. Additionally, layer normalization and nonlinear activation functions are incorporated within the 3×3 and 1×1 convolution modules to enhance the model's generalization ability and expressive power.

Residual connection: The processed contextual information is added to the original input through a residual connection, preserving the original features while incorporating global context. This fusion enhances feature representation and mitigates the gradient vanishing problem that may arise in deep networks, ensuring information integrity.

AECCM converts time series data from 1D to 2D, extracts global context through the spatial attention layer, captures local patterns through convolutional operations, and enhances feature representation via feature fusion. This makes AECCM particularly well-suited for capturing complex temporal dependencies and feature interactions in time series forecasting tasks.

3.4. Loss function and normalization

We adopt the Mean Absolute Error (MAE) as the loss function to evaluate the difference between predicted and actual values, with its definition shown below:

$$\mathrm{MAE} = \frac{1}{m}\sum_{i=1}^{m}\left|y_i - \hat{y}_i\right|. \tag{11}$$

where m is the number of samples, y_i is the true value of the predicted sequence at the ith time step, and $\hat{y}_i$ is the predicted result at the ith time step.

Additionally, we use Reversible Instance Normalization (RevIN) [36] to handle time series data, reducing distribution shift issues in time series forecasting. Specifically, we apply instance normalization before feeding the data into the model and perform denormalization on the results after prediction.

4. Experiments

4.1. Datasets and baselines

We evaluated the performance of our mLANet model using seven popular publicly available multivariate time series datasets, including four ETT datasets (ETTh1, ETTh2, ETTm1, ETTm2), Electricity, Traffic, and Weather. Detailed information about the experimental datasets is summarized in Table 1.

In recent years, deep learning models have made significant progress in time series forecasting, often outperforming traditional methods in various tasks. To comprehensively evaluate the forecasting performance of our mLANet model, we compare it with several state-of-the-art time series forecasting models, including APDNet [37], iTransformer [26], PatchTST [15], TimeMixer [38], TimesNet [29], GPT4TS [39], DLinear [12], FEDformer [20], ETSformer [11], PRformer [33], and P-sLSTM [40] . We use Mean Squared Error (MSE) and Mean Absolute Error (MAE) as the primary evaluation metrics, where lower values indicate higher forecasting accuracy. The formulas for these metrics are as follows:

$$\mathrm{MSE} = \frac{1}{N}\sum_{i=1}^{N}(t_i - p_i)^2, \mathrm{MAE} = \frac{1}{N}\sum_{i=1}^{N}|t_i - p_i| \tag{12}$$

where N is the total number of samples, t_i is the true value at the ith time step, and p_i is the predicted value at the ith time step.

4.2. Experimental results

To ensure a fair comparison, we adopted the mLANet model configuration in our experiments. For the voltage datasets (ETTh1, ETTh2, ETTm1, ETTm2), weather dataset, and traffic dataset, the historical lookback window length was set to 720. For the electricity dataset, the historical lookback window length was set to 660, and the prediction window length was chosen as $h \in \{96, 192, 336, 720\}$. The experimental results for PRformer and P-sLSTM were sourced from the best performance reported in their respective papers, while the results for other models were obtained from the best performance reported in APDNet [37]. In the experimental configuration of APDNet, the historical look-back window length was set to 96. The comparative experimental results are presented in Table 2.

In the experiments conducted on the ETTh datasets, we kept the parameters of other models unchanged, uniformly set the historical look-back window length to 720, and selected the prediction window length as $h \in 96, 192, 336, 720$. The related comparative experimental results are shown in Table 3.

To highlight mLANet's strength in capturing long sequence dependencies, we fixed the historical lookback window length to 720 across the ETT datasets and selected prediction window lengths $h \in \{96, 192, 336, 720\}$. All experiments were conducted using PyTorch on an NVIDIA GeForce RTX 4090 24 GB GPU.

Multivariate results

In the task of long-term multivariate time series forecasting, our mLANet achieved the best performance across seven benchmark datasets, as presented in Table 2. Compared to advanced models from 2024, our method (mLANet) reduced the average MSE by 8.89% and the average MAE by 5.60%. Specifically, compared to iTransformer from 2024, our method lowered the average MSE by 11.80% and the MAE by 7.93%. Against TimeMixer from 2024, mLANet reduced the average MSE by 13.14% and the MAE by 9.23%. When compared to the strongest baseline model from 2023, PatchTST, our approach decreased the average MSE by 17.35% and the MAE by 12.36%. Notably, on the ETTm1 and Weather datasets, mLANet's improvements were especially significant, with reductions in both average MSE and MAE exceeding 10%.

On the ETT datasets, we standardized the data input length to 720. The experimental results for prediction lengths of 96, 192, 336, and 720 are shown in Table 3, where mLANet continued to exhibit the best performance.

In conclusion, the experimental results demonstrate that mLANet achieves state-of-the-art performance in long-term multivariate time series forecasting tasks and is capable of effectively handling time series forecasting challenges across various domains.

4.3. Ablation studies

Ablation of mLANet

In this work, we propose a time series processing framework based on a recursive neural network, named mLANet. Compared to mainstream transformer-based frameworks for long-term multivariate time series prediction, mLANet is better at capturing long-sequence dependencies and reflecting temporal continuity and causality. Moreover, it is highly competitive in terms of computational efficiency and resource usage, with fewer model parameters, lower GPU resource requirements, and significantly reduced training and inference times. To validate the effectiveness of the mLANet architecture, we designed

Table 2
Multivariate long-term forecasting results on seven datasets with varying prediction lengths are shown below. Lower MSE or MAE indicates better forecasting. Best results are marked in red, second-best results are marked in blue, and ties are marked with the same color.

Models		mLANet (ours)		APDNet (2024)		iTransformer (2024)		PatchTST (2023)		TimeMixer (2024)		TimesNet (2023)		GPT4TS (2023)		DLinear (2023)		FEDformer (2022)		ETSformer (2023)		PRformer (2024)		P-sLSTM (2025)	
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	0.350	0.380	0.376	0.391	0.386	0.405	0.414	0.419	0.384	0.398	0.384	0.402	0.376	0.397	0.386	0.400	0.376	0.419	0.494	0.479	0.354	0.383	–	–
	192	0.392	0.407	0.431	0.422	0.441	0.436	0.460	0.445	0.439	0.429	0.436	0.429	0.438	0.426	0.437	0.432	0.420	0.448	0.538	0.504	0.397	0.410	–	–
	336	0.424	0.428	0.477	0.445	0.487	0.458	0.501	0.466	0.487	0.455	0.491	0.469	0.479	0.446	0.481	0.459	0.459	0.465	0.574	0.521	0.427	0.428	–	–
	720	0.446	0.464	0.488	0.472	0.503	0.491	0.500	0.488	0.502	0.482	0.521	0.500	0.495	0.476	0.519	0.516	0.506	0.507	0.562	0.562	0.489	0.492	–	–
ETTh2	96	0.266	0.325	0.290	0.340	0.297	0.349	0.302	0.348	0.293	0.344	0.340	0.374	0.295	0.348	0.333	0.387	0.358	0.397	0.340	0.391	0.268	0.327	–	–
	192	0.333	0.371	0.369	0.390	0.380	0.400	0.388	0.400	0.374	0.396	0.402	0.414	0.386	0.404	0.477	0.476	0.429	0.439	0.430	0.439	0.332	0.370	–	–
	336	0.363	0.396	0.411	0.424	0.428	0.432	0.426	0.433	0.416	0.430	0.452	0.452	0.421	0.435	0.594	0.541	0.496	0.487	0.485	0.479	0.361	0.395	–	–
	720	0.392	0.427	0.421	0.439	0.427	0.445	0.431	0.446	0.440	0.452	0.462	0.468	0.422	0.445	0.831	0.657	0.463	0.474	0.500	0.497	0.396	0.429	–	–
ETTm1	96	0.280	0.331	0.315	0.354	0.334	0.368	0.334	0.372	0.333	0.371	0.338	0.375	0.329	0.364	0.345	0.372	0.379	0.419	0.375	0.398	0.278	0.333	0.292	0.343
	192	0.321	0.357	0.360	0.378	0.377	0.391	0.378	0.394	0.367	0.386	0.374	0.387	0.368	0.382	0.380	0.389	0.426	0.441	0.408	0.410	0.324	0.361	0.329	0.369
	336	0.354	0.378	0.392	0.401	0.426	0.420	0.406	0.414	0.397	0.408	0.410	0.411	0.400	0.403	0.413	0.413	0.445	0.459	0.435	0.428	0.362	0.384	0.362	0.391
	720	0.411	0.409	0.456	0.439	0.491	0.459	0.462	0.445	0.460	0.445	0.478	0.450	0.460	0.439	0.474	0.453	0.543	0.490	0.499	0.462	0.426	0.425	0.421	0.424
ETTm2	96	0.154	0.239	0.171	0.253	0.180	0.264	0.175	0.259	0.174	0.258	0.187	0.267	0.178	0.263	0.193	0.292	0.203	0.287	0.189	0.280	0.162	0.245	–	–
	192	0.210	0.280	0.235	0.295	0.250	0.309	0.240	0.302	0.239	0.302	0.249	0.309	0.245	0.306	0.284	0.362	0.269	0.328	0.253	0.319	0.219	0.286	–	–
	336	0.272	0.321	0.292	0.333	0.311	0.348	0.302	0.342	0.296	0.340	0.321	0.351	0.309	0.347	0.369	0.427	0.325	0.366	0.314	0.357	0.272	0.326	–	–
	720	0.352	0.375	0.388	0.389	0.412	0.407	0.399	0.397	0.393	0.397	0.408	0.403	0.409	0.408	0.554	0.522	0.421	0.415	0.414	0.413	0.359	0.383	–	–
Electricity	96	0.130	0.224	0.141	0.239	0.148	0.240	0.195	0.285	0.157	0.250	0.168	0.272	0.185	0.272	0.197	0.282	0.193	0.308	0.187	0.304	0.127	0.217	0.130	0.226
	192	0.153	0.244	0.158	0.252	0.162	0.253	0.199	0.289	0.171	0.263	0.184	0.289	0.189	0.276	0.196	0.285	0.201	0.315	0.199	0.315	0.148	0.237	0.148	0.243
	336	0.166	0.259	0.174	0.271	0.178	0.269	0.215	0.305	0.189	0.282	0.198	0.300	0.204	0.291	0.209	0.301	0.214	0.329	0.212	0.329	0.161	0.252	0.165	0.262
	720	0.190	0.282	0.207	0.302	0.225	0.317	0.256	0.337	0.233	0.316	0.220	0.320	0.245	0.324	0.245	0.333	0.246	0.355	0.233	0.345	0.185	0.275	0.199	0.293
Weather	96	0.139	0.179	0.159	0.205	0.174	0.214	0.177	0.218	0.166	0.213	0.172	0.220	0.182	0.223	0.196	0.255	0.217	0.296	0.197	0.281	0.144	0.187	0.149	0.208
	192	0.185	0.224	0.207	0.250	0.221	0.254	0.225	0.259	0.209	0.251	0.219	0.261	0.231	0.263	0.237	0.296	0.276	0.336	0.237	0.312	0.188	0.233	0.197	0.256
	336	0.235	0.266	0.263	0.291	0.278	0.296	0.278	0.297	0.264	0.293	0.280	0.306	0.283	0.300	0.283	0.335	0.339	0.380	0.298	0.353	0.241	0.274	0.249	0.297
	720	0.304	0.316	0.345	0.343	0.358	0.349	0.354	0.348	0.342	0.343	0.365	0.359	0.360	0.350	0.345	0.381	0.403	0.428	0.352	0.388	0.326	0.332	0.320	0.350
Traffic	96	0.367	0.245	0.380	0.265	0.395	0.268	0.544	0.359	0.485	0.323	0.593	0.321	0.468	0.307	0.650	0.396	0.587	0.366	0.607	0.392	0.353	0.222	–	–
	192	0.392	0.257	0.414	0.276	0.417	0.276	0.540	0.354	0.488	0.322	0.617	0.336	0.476	0.311	0.598	0.370	0.604	0.373	0.621	0.399	0.372	0.233	–	–
	336	0.408	0.265	0.432	0.283	0.433	0.283	0.551	0.358	0.507	0.321	0.629	0.336	0.488	0.317	0.605	0.373	0.621	0.383	0.622	0.396	0.385	0.241	–	–
	720	0.448	0.285	0.461	0.300	0.467	0.302	0.586	0.375	0.549	0.335	0.640	0.350	0.521	0.333	0.645	0.394	0.626	0.382	0.632	0.396	0.421	0.258	–	–

Table 3
On the ETT dataset, the long-term multivariate time series forecasting results with an input length of 720 and prediction lengths of {96, 192, 336, 720}.

Models		mLANet (ours)		APDNet (2024)		iTransformer (2024)		PatchTST (2023)		TimeMixer (2024)		TimesNet (2023)		GPT4TS (2023)		DLinear (2023)		FEDformer (2022)		ETSformer (2023)		PRformer (2024)		P-sLSTM (2025)	
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	0.350	0.380	0.381	0.409	0.392	0.424	0.426	0.439	0.414	0.440	0.438	0.454	0.375	0.407	0.369	0.399	0.482	0.498	0.523	0.532	0.356	0.384	0.381	0.414
	192	0.392	0.407	0.419	0.434	0.437	0.455	0.466	0.469	0.441	0.454	0.455	0.469	0.413	0.429	0.405	0.422	0.483	0.501	0.581	0.570	0.397	0.410	0.424	0.444
	336	0.424	0.428	0.450	0.456	0.457	0.470	0.531	0.512	0.524	0.506	0.495	0.494	0.430	0.441	0.440	0.452	0.493	0.500	0.634	0.605	0.428	0.429	0.479	0.479
	720	0.446	0.464	0.475	0.476	0.553	0.537	0.525	0.515	0.774	0.628	0.636	0.580	0.449	0.466	0.488	0.507	0.642	0.594	0.629	0.603	0.504	0.499	0.548	0.531
ETTh2	96	0.266	0.325	0.291	0.349	0.303	0.364	0.339	0.383	0.293	0.364	0.442	0.457	0.290	0.348	0.304	0.370	0.409	0.456	0.400	0.460	0.270	0.328	0.389	0.427
	192	0.333	0.371	0.354	0.391	0.410	0.423	0.399	0.420	0.371	0.410	0.414	0.431	0.360	0.396	0.386	0.423	0.420	0.462	0.444	0.486	0.334	0.369	0.610	0.550
	336	0.363	0.396	0.409	0.430	0.440	0.450	0.505	0.472	0.373	0.416	0.411	0.440	0.381	0.419	0.539	0.512	0.443	0.481	0.477	0.510	0.363	0.396	0.688	0.588
	720	0.392	0.427	0.462	0.471	0.439	0.469	0.538	0.515	27.667	3.429	0.591	0.536	0.443	0.467	0.864	0.658	0.552	0.540	0.555	0.554	0.383	0.419	1.312	0.804
ETTm1	96	0.280	0.331	0.304	0.357	0.319	0.367	0.294	0.353	0.321	0.362	0.356	0.392	0.291	0.351	0.306	0.349	0.363	0.422	0.568	0.533	0.273	0.325	0.293	0.348
	192	0.321	0.357	0.340	0.379	0.347	0.388	0.353	0.397	0.342	0.378	0.494	0.452	0.332	0.375	0.335	0.366	0.401	0.445	0.654	0.578	0.316	0.352	0.342	0.379
	336	0.354	0.378	0.367	0.395	0.387	0.413	0.371	0.405	0.374	0.399	0.430	0.437	0.365	0.391	0.374	0.398	0.422	0.447	0.682	0.589	0.349	0.373	0.373	0.400
	720	0.411	0.409	0.436	0.431	0.437	0.439	0.431	0.444	0.436	0.432	0.486	0.480	0.420	0.422	0.414	0.414	0.505	0.491	0.742	0.629	0.405	0.406	0.422	0.427
ETTm2	96	0.154	0.239	0.178	0.262	0.180	0.273	0.185	0.282	0.221	0.313	0.209	0.290	0.176	0.264	0.162	0.256	0.299	0.362	0.221	0.332	0.159	0.242	0.192	0.284
	192	0.210	0.280	0.235	0.302	0.243	0.316	0.243	0.321	0.241	0.313	0.297	0.346	0.241	0.309	0.221	0.300	0.323	0.375	0.304	0.387	0.218	0.286	0.289	0.353
	336	0.272	0.321	0.301	0.343	0.299	0.352	0.296	0.352	0.290	0.345	0.339	0.380	0.270	0.327	0.282	0.345	0.373	0.413	0.434	0.475	0.278	0.329	0.323	0.377
	720	0.352	0.375	0.366	0.389	0.382	0.405	0.432	0.429	0.371	0.400	0.375	0.404	0.380	0.404	0.402	0.427	0.441	0.455	0.792	0.663	0.356	0.382	0.455	0.459

Table 4
Ablation study of mLANet, where mLANet-transformer is used to replace mLANet.

Models		mLANet						mLANet-transformer					
Metric		MSE	MAE	Parameters	Training-epoch	Inference-batch	GPU-memory	MSE	MAE	Parameters	Training-epoch	Inference-batch	GPU-memory
ETTh1	96	0.350	0.380	1.64M	1.04 s	2.2 ms	1634 MiB	0.356	0.384	16.92M	3.37 s	5.5 ms	2812 MiB
	192	0.392	0.407	1.69M	1.04 s	2.2 ms	1630 MiB	0.397	0.410	16.99M	2.34 s	3.2 ms	3438 MiB
	336	0.425	0.429	1.76M	1.01 s	2.2 ms	1632 MiB	0.428	0.429	17.09M	1.94 s	3.4 ms	3460 MiB
	720	0.446	0.465	1.96M	1.08 s	2.2 ms	1628 MiB	0.504	0.499	17.37M	1.90 s	3.0 ms	3456 MiB
ETTm2	96	0.154	0.239	1.69M	7.17 s	2.2 ms	9758 MiB	0.162	0.244	4.44M	11.81 s	2.3 ms	14 824 MiB
	192	0.210	0.280	1.74M	6.77 s	2.2 ms	9756 MiB	0.224	0.290	4.49M	11.59 s	5.4 ms	14 842 MiB
	336	0.272	0.321	1.81M	6.71 s	2.1 ms	9756 MiB	0.276	0.328	4.56M	16.90 s	5.4 ms	14 842 MiB
	720	0.352	0.375	2.01M	6.77 s	2.2 ms	9752 MiB	0.359	0.381	4.76M	16.02 s	5.4 ms	14 840 MiB
Weather	96	0.139	0.179	1.69M	24.27 s	3.4 ms	5058 MiB	0.142	0.184	6.23M	108.52 s	58.7 ms	5974 MiB
	192	0.185	0.224	1.74M	23.94 s	3.6 ms	5060 MiB	0.191	0.233	6.28M	95.63 s	56.6 ms	5974 MiB
	336	0.235	0.266	1.81M	29.89 s	5.5 ms	5056 MiB	0.242	0.273	6.35M	65.96 s	11.5 ms	5978 MiB
	720	0.304	0.317	2.01M	45.59 s	4.8 ms	5076 MiB	0.316	0.325	6.55M	39.44 s	9.5 ms	5972 MiB

a variant, mLANet-transformer, by replacing the mLANet architecture with the encoder of Transformer, and conducted comparative experiments. As shown in Table 4, the experimental results demonstrate that mLANet significantly outperforms mLANet-transformer in terms of mean squared error (MSE), mean absolute error (MAE), model parameter count, GPU resource consumption, average training time per epoch, and inference time. These results fully demonstrate the superior performance and efficiency of mLANet in long-term multivariate time series prediction.

Ablation of AECCM in 2D

This paper introduces a method that transforms one-dimensional time series data into a two-dimensional representation by splitting the sequence along the time dimension, and then applies 2D convolutional neural networks to perform feature extraction. To validate the effectiveness of this module, we conducted an ablation experiment by omitting the transformation from 1D to 2D in the AECCM module and substituting 1D convolutions for all 2D convolution operations. The results are shown in Table 5. The results indicate that the AECCM2D module enhances the model's ability to extract features from long

Table 5
The ablation study of the AECCM2D in this model modifies mLANet into mLA1DNet*, a variant where the AECCM is adapted to a 1D version.

Datasets		ETTh1				ETTm2				Weather			
Prediction length		96	192	336	720	96	192	336	720	96	192	336	720
mLANet	MSE	**0.350**	**0.392**	**0.424**	0.446	**0.154**	**0.210**	0.272	0.352	**0.139**	**0.185**	**0.235**	**0.304**
	MAE	**0.380**	**0.408**	**0.429**	0.465	**0.239**	**0.280**	**0.321**	0.375	**0.179**	**0.224**	**0.266**	**0.317**
mLA1DNet*	MSE	0.355	0.398	0.430	**0.444**	0.156	**0.210**	**0.268**	0.369	0.142	**0.185**	0.241	0.313
	MAE	0.382	0.410	0.433	**0.464**	0.241	0.281	**0.321**	0.389	0.183	0.225	0.272	0.322

Table 6
The ablation study of the AECCM in this model removes the AECCM module from mLANet to create mLNet*.

Datasets		ETTh1				ETTm2				Weather			
Prediction length		96	192	336	720	96	192	336	720	96	192	336	720
mLANet	MSE	**0.350**	**0.392**	**0.424**	0.446	**0.154**	0.210	0.272	0.352	**0.139**	**0.185**	**0.235**	**0.304**
	MAE	0.380	**0.408**	**0.429**	0.465	**0.239**	0.280	0.321	0.375	**0.179**	**0.224**	**0.266**	**0.317**
mLNet*	MSE	**0.350**	0.399	0.431	0.465	**0.154**	**0.208**	**0.265**	**0.347**	0.142	0.186	0.240	0.329
	MAE	**0.379**	0.412	0.434	0.476	0.241	**0.279**	**0.318**	**0.373**	0.183	0.227	0.270	0.333

Table 7
Impact of the number of channels. Set $C \in \{2, 4, 8, 16\}$ and evaluate MSE and MAE across various prediction lengths, where lower values indicate better performance.

Datasets		ETTm2				Weather			
Prediction length		96	192	336	720	96	192	336	720
C = 2	MSE	0.154	0.221	0.259	0.342	0.142	0.187	0.238	0.335
	MAE	0.241	0.286	0.314	0.371	0.184	0.228	0.269	0.338
C = 4	MSE	0.157	0.210	0.261	0.343	0.140	0.186	0.237	0.325
	MAE	0.240	0.279	0.314	0.371	0.180	0.226	0.267	0.328
C = 8	MSE	0.154	0.210	0.272	0.352	0.139	0.185	0.235	0.304
	MAE	0.239	0.280	0.321	0.375	0.179	0.224	0.266	0.317
C = 16	MSE	0.156	0.208	0.264	0.344	0.143	0.185	0.235	0.314
	MAE	0.240	0.278	0.318	0.373	0.182	0.225	0.266	0.324

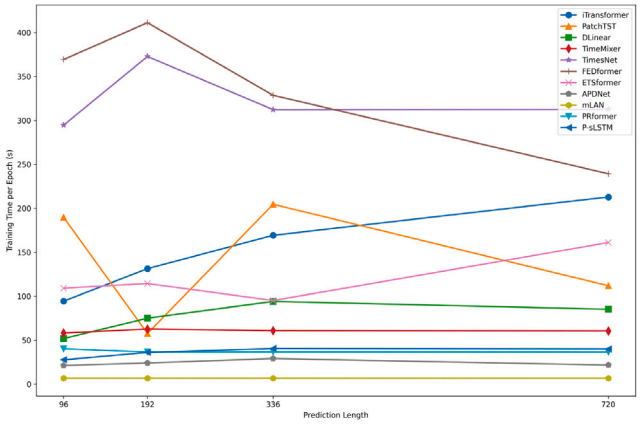

Fig. 4. Training on the ETTm2 dataset with the input-720-predict-{96,192,336,720} configuration, and comparing the training time with other baseline models.

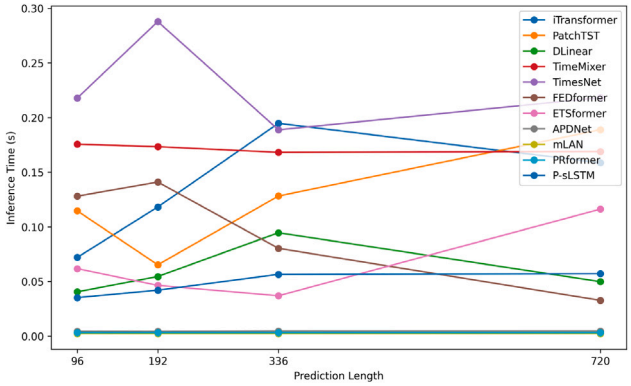

Fig. 5. Inference on the ETTm2 dataset with the input-720-predict-{96,192,336,720} configuration, and comparing the inference time with other baseline models.

sequences, thus improving the prediction performance in time series forecasting.

Ablation of AECCM

To evaluate the importance of the AECCM module in the mLAN-Net architecture, we conducted ablation experiments to compare the performance of models with and without the AECCM module in long-term multivariate time series forecasting tasks. The experimental results, as shown in Table 6, demonstrate that the AECCM module significantly enhances the model's prediction accuracy and robustness, making it a critical component for mLAN-Net to achieve outstanding performance. In particular, on the ETTh1 and Weather datasets, the AECCM module improves the model's ability to capture local patterns, global context, and long-term dependencies, thereby greatly enhancing its capacity to model complex time series data. These datasets feature more pronounced long-term trends and intricate temporal dependencies, areas where the AECCM module demonstrates distinct advantages in handling such complexities. Therefore, the design of the AECCM module—particularly its two-dimensional transformation, attention mechanism, and feature fusion strategy—enables the model to more effectively capture underlying patterns in multivariate time series. This provides robust support for mLAN-Net to deliver leading performance in long-term time series forecasting tasks.

4.4. Runtime and computational cost

To obtain detailed information such as model parameters, average training time per epoch, and inference time, we conducted experiments

Table 8
Impact of the convolution kernel size. Set $K \in \{3, 5, 7, 9, 11\}$ and evaluate MSE and MAE across various prediction lengths, where lower values indicate better performance.

Datasets		ETTm2				Weather			
Prediction length		96	192	336	720	96	192	336	720
K = 3	MSE	0.153	0.209	0.261	0.354	0.142	0.186	0.236	0.330
	MAE	0.239	0.279	0.316	0.378	0.183	0.227	0.267	0.333
K = 5	MSE	0.153	0.207	0.261	0.341	0.140	0.186	0.238	0.324
	MAE	0.239	0.279	0.315	0.370	0.180	0.227	0.266	0.328
K = 7	MSE	0.154	0.215	0.262	0.381	0.142	0.186	0.236	0.314
	MAE	0.240	0.283	0.315	0.393	0.181	0.225	0.267	0.324
K = 9	MSE	0.154	0.210	0.272	0.352	0.139	0.185	0.235	0.304
	MAE	0.239	0.280	0.321	0.375	0.179	0.224	0.265	0.316
K = 11	MSE	0.154	0.207	0.264	0.356	0.140	0.184	0.237	0.313
	MAE	0.239	0.278	0.316	0.380	0.181	0.225	0.268	0.322

Table 9
In our noise injection study, we incorporated Gaussian noise with a standard deviation of $0.1 \cdot \mathcal{N}(0, 1)$ during training and testing. The experiments were conducted on the ETTh1 dataset with a historical look-back window size of 720, and prediction lengths $O \in \{96, 192, 336, 720\}$.

Training	Without noise				With noise			
Testing	Without noise		With noise		Without noise		With noise	
Metrics	MSE	MAE	MSE	MAE	MSE	MAE	MSE	MAE
ETTh1	0.350	0.380	0.351	0.381	0.350	0.380	0.352	0.382
	0.392	0.407	0.393	0.408	0.393	0.408	0.394	0.409
	0.424	0.428	0.425	0.429	0.422	0.426	0.422	0.426
	0.446	0.464	0.447	0.465	0.448	0.466	0.449	0.467

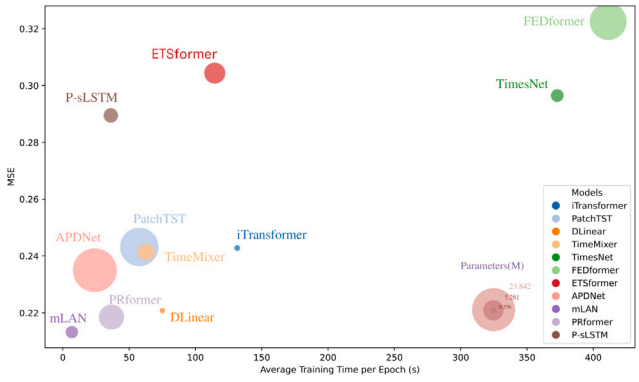

Fig. 6. Model efficiency comparison under input-720-predict-192 of ETTm2.

on the ETTm2 dataset with the input length fixed at 720 and the prediction length fixed at 192. We compared the training and inference times of the proposed method with those of other methods. As shown in Figs. 4 and 5, the average training and inference times per epoch of mLANet are lower than those of advanced methods such as PatchTST, DLinear, APDNet, TimeMixer, iTransformer, FEDformer, and TimesNet. Furthermore, as illustrated in Fig. 6, in almost all cases, mLANet exhibits the fastest inference speed, the fewest number of parameters, and the highest prediction accuracy. These results indicate that mLANet is highly applicable and practical for long-term time series forecasting tasks, making it a very efficient method.

4.5. Visualization of forecast results

In the ETTm2 dataset, we conducted visualization experiments on different models with an input length of 720 and prediction horizons $h \in \{96, 192, 336, 720\}$ (see Figs. 7, 8, 9, and 10). The experimental results show that, compared to methods such as PatchTST, DLinear, APDNet, TimeMixer, iTransformer, PRformer, P-sLSTM, and mLANet exhibits the best generalization ability across multiple prediction horizons.

4.6. Hyperparameter sensitivity

To investigate the impact of the number of segmentation channels C in the AECCM module and the one-dimensional convolution kernel size K in the mLSTM module on time series forecasting tasks, we conducted a sensitivity analysis on the ETTm2 and Weather datasets. First, we fixed the input sequence length at 720, kept other model parameters constant, and varied the number of segmentation channels $C \in \{2, 4, 8, 16\}$, evaluating the mean squared error (MSE) and mean absolute error (MAE) for different prediction lengths. As shown in Table 7, on the ETTm2 dataset (with fewer features), the number of channels has no significant impact on model performance for prediction lengths of 96 and 192; for prediction lengths of 336 and 720, smaller channel numbers yield better performance. On the Weather dataset (with more features), the impact of channel numbers is more pronounced for a prediction length of 720, but not significant for other lengths (96, 192, 336). Next, we fixed the input sequence length at 720, kept other parameters constant, and varied $K \in \{3, 5, 7, 9, 11\}$, evaluating MSE and MAE for different prediction lengths. As shown in Table 8, the choice of kernel size K affects model performance for a prediction length of 720; for other prediction lengths, the impact of K is minimal, with overall performance remaining stable.

4.7. Robustness analysis

To simulate real-world applications, we introduced common sensor noise by injecting $0.1 \cdot \mathcal{N}(0, 1)$ Gaussian noise into the raw data during both training and testing phases to demonstrate the robustness of our model. Specifically, we designed four sets of experiments: the first set involves adding noise during the training phase, with testing conducted under both noisy and noise-free conditions; the second set involves no noise during the training phase, with testing conducted under both noisy and noise-free conditions. Experiments were performed on the ETTh1 dataset. As shown in Table 9, adding Gaussian noise during

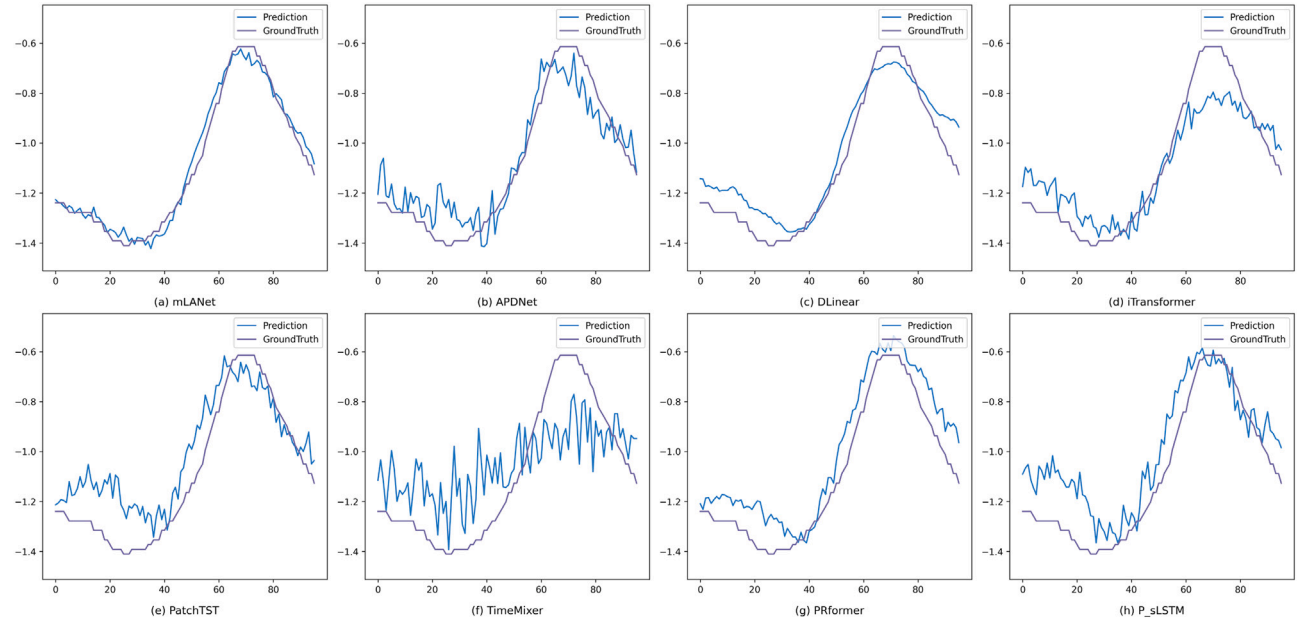

Fig. 7. Prediction from the ETTm2 dataset with the input-720-predict-96 configuration.

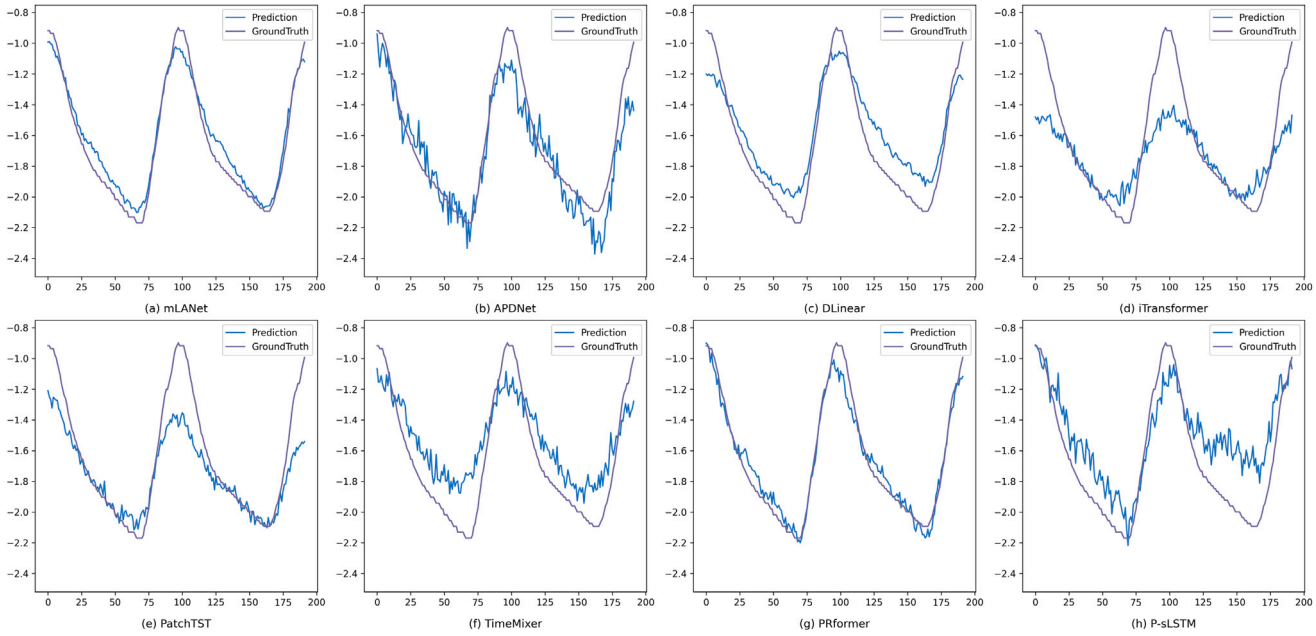

Fig. 8. Prediction from the ETTm2 dataset with the input-720-predict-192 configuration.

both training and testing phases has a limited impact on our model's performance. Even in the worst case, the evaluation metrics MSE and MAE increase by only 0.003. Furthermore, when the model is trained on noisy data and predicts on both noise-free and noisy test data (with a prediction length of 336), its MSE and MAE decrease by 0.002 compared to the noise-free scenario, further demonstrating the model's robustness. Notably, the model maintains consistent robustness across different prediction lengths, reinforcing its stability.

5. Conclusion

This paper presents an innovative time series forecasting model framework, mLAN-Net, designed to address the challenge of capturing long-term dependencies. At its core, mLAN-Net leverages a variant of recurrent neural networks, the mLSTM module, which significantly enhances memory capacity by stacking multiple layers of matrix memory units, effectively capturing dependencies over long time spans. Additionally, the introduced AECCM module transforms one-dimensional time series into two-dimensional representations, combining convolutional neural networks and attention mechanisms to simultaneously extract both local and global features. This approach improves modeling accuracy through selective feature fusion. Furthermore, our proposed time series transformation strategy processes multiple time windows in parallel, reducing computational complexity and strengthening the ability to capture local patterns. Experimental results demonstrate that mLAN-Net achieves state-of-the-art forecasting performance across seven public multivariate time series datasets, outperforming current leading methods in terms of accuracy and robustness. This achievement highlights the superiority of mLAN-Net in long-sequence forecasting

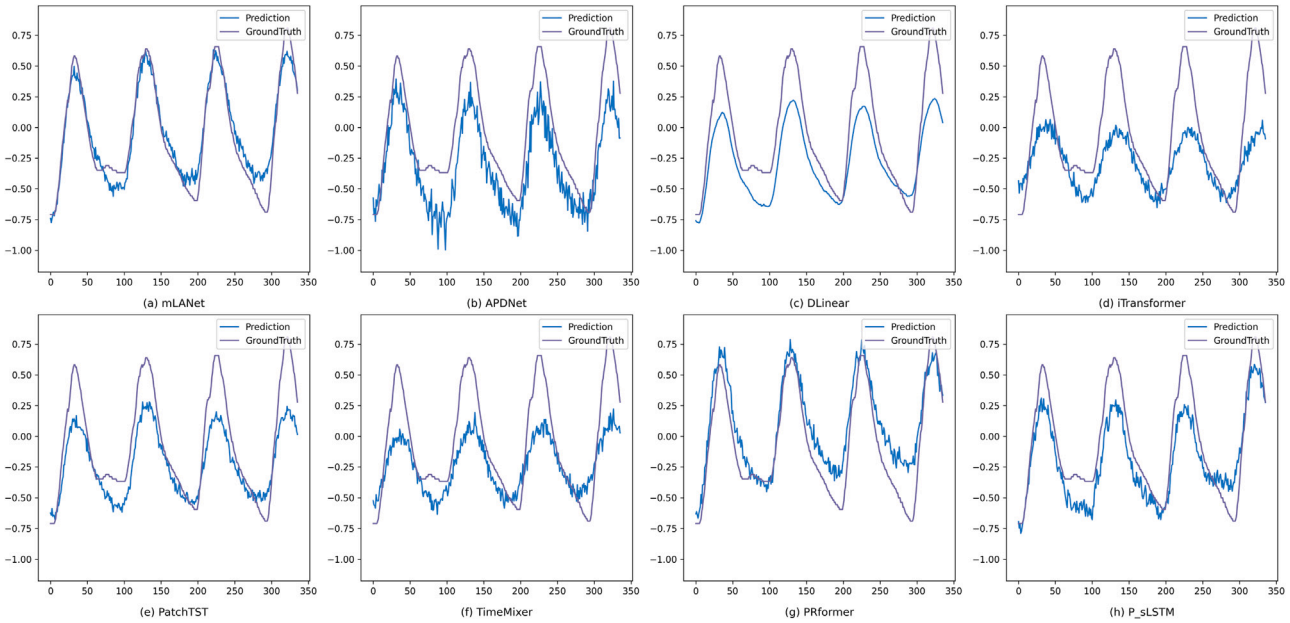

Fig. 9. Prediction from the ETTm2 dataset with the input-720-predict-336 configuration.

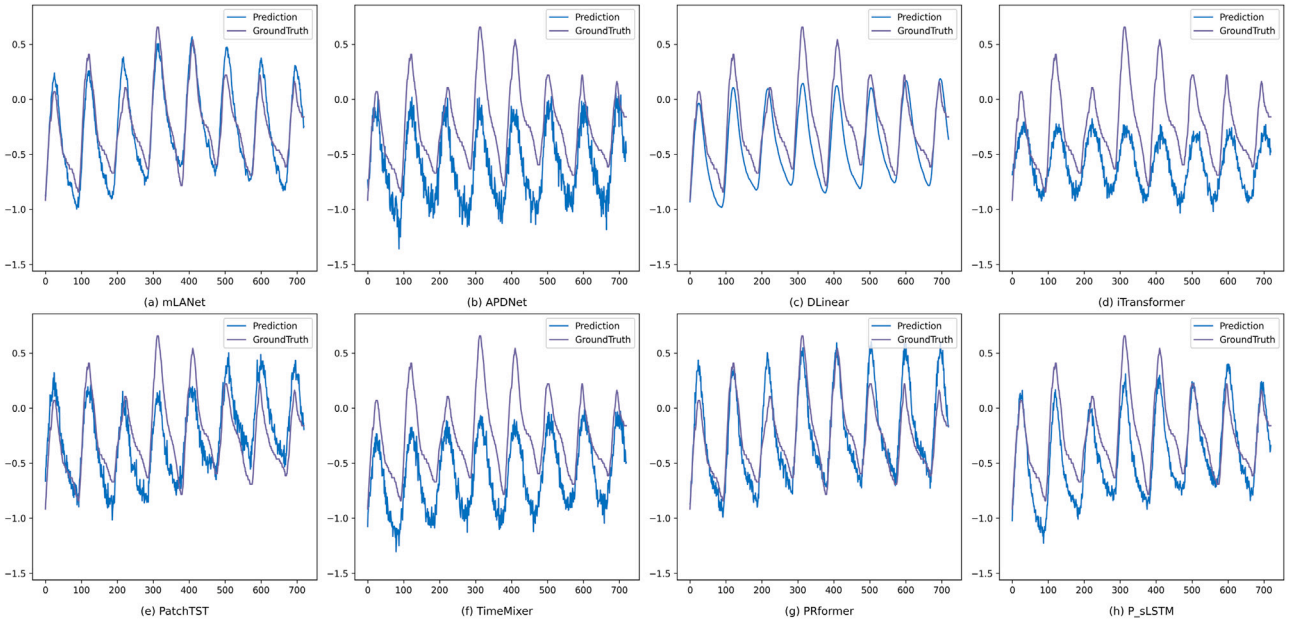

Fig. 10. Prediction from the ETTm2 dataset with the input-720-predict-720 configuration.

and provides a powerful solution for the time series domain. We believe that the advantages of mLAN-Net will drive its widespread adoption in long-term multivariate time series forecasting applications and explore its potential in more diverse scenarios.

CRediT authorship contribution statement

Jihua Jiang: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Investigation. **Bin Jiang:** Supervision, Project administration, Conceptualization. **Yong Wang:** Writing – review & editing, Supervision, Resources. **Duoqian Miao:** Funding acquisition, Formal analysis.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the School of Computer Science and Technology, Tongji University, China, and the National Natural Science Foundation of China (Grant No. 62376198).

Data availability

In the manuscript, I have provided the GitHub link to the data and code.

References

[1] G.U. Yule, On a method of investigating periodicities in disturbed series with special reference to Wolfer's sunspot numbers, in: Statistical Papers of George Udny Yule, Vol. 226, Hafner Press New York, 1971, pp. 389–420.

[2] G.T. Walker, On periodicity in series of related terms, Proc. R. Soc. Lond. Ser. A Math. Phys. Eng. Sci. 131 (818) (1931) 518–532.

[3] I. Rojas, O. Valenzuela, F. Rojas, A. Guillén, L.J. Herrera, H. Pomares, L. Marquez, M. Pasadas, Soft-computing techniques and ARMA model for time series prediction, Neurocomputing 71 (4–6) (2008) 519–537.

[4] M. Valipour, Critical areas of Iran for agriculture water management according to the annual rainfall, Eur. J. Sci. Res. 84 (4) (2012) 600–608.

[5] Z. Cui, K. Henrickson, R. Ke, Y. Wang, Traffic graph convolutional recurrent neural network: A deep learning framework for network-scale traffic learning and forecasting, IEEE Trans. Intell. Transp. Syst. 21 (11) (2019) 4883–4894.

[6] L. Zhu, W. Wen, J. Li, C. Zhang, Y. Shen, Y. Hou, T. Liu, Structure-aware recurrent learning machine for short-term Voltage trajectory sensitivity prediction, IEEE Internet Things J. 11 (9) (2023) 15128–15139.

[7] X. Wang, G.Q. Ma, A. Eden, C. Li, A. Trott, S. Zheng, D. Parkes, Platform behavior under market shocks: A simulation framework and reinforcement-learning based study, in: Proceedings of the ACM Web Conference 2023, 2023, pp. 3592–3602.

[8] S.N. Ward, Area-based tests of long-term seismic hazard predictions, Bull. Seismol. Soc. Am. 85 (5) (1995) 1285–1298.

[9] C. Puri, G. Kooijman, B. Vanrumste, S. Luca, Forecasting time series in healthcare with Gaussian processes and dynamic time warping based subset selection, IEEE J. Biomed. Heal. Inform. 26 (12) (2022) 6126–6137.

[10] H. Zhou, S. Zhang, J. Peng, S. Zhang, J. Li, H. Xiong, W. Zhang, Informer: Beyond efficient transformer for long sequence time-series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 35, 2021, pp. 11106–11115.

[11] G. Woo, C. Liu, D. Sahoo, A. Kumar, S. Hoi, Etsformer: Exponential smoothing transformers for time-series forecasting, in: Proc. Int. Conf. Learn. Represent., 2023, pp. 1–19.

[12] A. Zeng, M. Chen, L. Zhang, Q. Xu, Are transformers effective for time series forecasting? in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 37, 2023, pp. 11121–11128.

[13] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, Adv. Neural Inf. Process. Syst. 30 (2017).

[14] Y. Zhang, R. Wu, S.M. Dascalu, F.C. Harris Jr., Sparse transformer with local and seasonal adaptation for multivariate time series forecasting, Sci. Rep. 14 (1) (2024) 15909.

[15] Y. Nie, N.H. Nguyen, P. Sinthong, J. Kalagnanam, A time series is worth 64 words: Long-term forecasting with transformers, in: Proc. 11th Int. Conf. Learn. Represent., 2023, pp. 1–24.

[16] L.R. Medsker, L. Jain, et al., Recurrent neural networks, Des. Appl. 5 (64–67) (2001) 2.

[17] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.

[18] J. Chung, C. Gulcehre, K. Cho, Y. Bengio, Empirical evaluation of gated recurrent neural networks on sequence modeling, 2014, arXiv preprint arXiv:1412.3555.

[19] Y. Bengio, P. Simard, P. Frasconi, Learning long-term dependencies with gradient descent is difficult, IEEE Trans. Neural Netw. 5 (2) (1994) 157–166.

[20] T. Zhou, Z. Ma, Q. Wen, X. Wang, L. Sun, R. Jin, Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting, in: International Conference on Machine Learning, PMLR, 2022, pp. 27268–27286.

[21] K. Pöppel, M. Beck, M. Spanring, A. Auer, O. Prudnikova, M.K. Kopp, G. Klambauer, J. Brandstetter, S. Hochreiter, Xlstm: Extended long short-term memory, in: First Workshop on Long-Context Foundation Models@, ICML 2024, 2024.

[22] S. Lin, W. Lin, W. Wu, F. Zhao, R. Mo, H. Zhang, Segrnn: Segment recurrent neural network for long-term time series forecasting, 2023, arXiv preprint arXiv:2308.11200.

[23] G. Lai, W.-C. Chang, Y. Yang, 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, pp. 95–104.

[24] S. Li, X. Jin, Y. Xuan, X. Zhou, W. Chen, Y.-X. Wang, X. Yan, Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting, Adv. Neural Inf. Process. Syst. 32 (2019).

[25] H. Wu, J. Xu, J. Wang, M. Long, Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting, Adv. Neural Inf. Process. Syst. 34 (2021) 22419–22430.

[26] Y. Liu, T. Hu, H. Zhang, H. Wu, S. Wang, L. Ma, M. Long, Itransformer: Inverted transformers are effective for time series forecasting, 2023, arXiv preprint arXiv:2310.06625.

[27] S.-A. Chen, C.-L. Li, N. Yoder, S.O. Arik, T. Pfister, Tsmixer: An all-mlp architecture for time series forecasting, 2023, arXiv preprint arXiv:2303.06053.

[28] H. Wang, J. Peng, F. Huang, J. Wang, J. Chen, Y. Xiao, "MICN: Multi-scale local and global context modeling for long-term series forecasting, in: Proc. Int. Conf. Learn. Represent., 2023, pp. 1–22.

[29] H. Wu, T. Hu, Y. Liu, H. Zhou, J. Wang, M. Long, TimesNet: Temporal 2D-variation modeling for general time series analysis, in: Proc. Int. Conf. Learn. Represent., 2023, pp. 1–23.

[30] Z. Wang, T. Oates, Imaging time-series to improve classification and imputation, in: Proc. Int. Joint Conf. Artif. Intell., 2015, pp. 3939–3945.

[31] Z. Wang, T. Oates, et al., Encoding time series as images for visual inspection and classification using tiled convolutional neural networks, in: Workshops At the Twenty-Ninth AAAI Conference on Artificial Intelligence, Vol. 1, Austin, TX, 2015, pp. 1–7.

[32] Y. Wu, K. He, Group normalization, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 3–19.

[33] Y. Yu, W. Yu, F. Nie, X. Li, PRformer: Pyramidal recurrent transformer for multivariate time series forecasting, 2024, arXiv preprint arXiv:2408.10483.

[34] T.J. Sejnowski, Storing covariance with nonlinearly interacting neurons, J. Math. Biol. 4 (4) (1977) 303–321.

[35] P. Dayan, D.J. Willshaw, Optimising synaptic learning rules in linear associative memories, Biol. Cybernet. 65 (4) (1991) 253–265.

[36] T. Kim, J. Kim, Y. Tae, C. Park, J.-H. Choi, J. Choo, Reversible instance normalization for accurate time-series forecasting against distribution shift, in: Proc. Int. Conf. Learn. Represent., 2022, pp. 1–25.

[37] W. Zhuang, J. Fan, J. Fang, W. Fang, M. Xia, Rethinking general time series analysis from a frequency domain perspective, Knowl.-Based Syst. 301 (2024) 112281.

[38] S. Wang, H. Wu, X. Shi, T. Hu, H. Luo, L. Ma, J.Y. Zhang, J. Zhou, Timemixer: Decomposable multiscale mixing for time series forecasting, in: Proc. Int. Conf. Learn. Represent., 2024, pp. 1–27.

[39] T. Zhou, P. Niu, L. Sun, R. Jin, et al., One fits all: Power general time series analysis by pretrained lm, Adv. Neural Inf. Process. Syst. 36 (2023) 43322–43355.

[40] Y. Kong, Z. Wang, Y. Nie, T. Zhou, S. Zohren, Y. Liang, P. Sun, Q. Wen, Unlocking the power of lstm for long term time series forecasting, in: Proceedings of the AAAI Conference on Artificial Intelligence, Vol. 39, 2025, pp. 11968–11976.