LAMEE: A Light All-MLP Framework for TimeSeries Prediction Empowering Recommendations

Yi Xie
Shanghai Key Lab of Data Science, Fudan University, Shanghai
Yun Xiong
Shanghai Key Lab of Data Science, Fudan University, Shanghai
Xiaofeng Gao (gao-xf@cs.stju.edu.cn)
MoE Key Lab of Artificial Intelligence, Shanghai Jiaotong University
Jiadong Chen
MoE Key Lab of Artificial Intelligence, Shanghai Jiaotong University
Yao Zhang
Shanghai Key Lab of Data Science, Fudan University, Shanghai
Xian Wu
Shanghai Key Lab of Data Science, Fudan University, Shanghai
Chao Chen
College of Computer Science, Chongqing University

Research Article

Keywords: time series analysis, multi-layer perceptron, joint time-frequency information, lightweight framework, recommendation systems

Posted Date: December 21st, 2023

DOI: https://doi.org/10.21203/rs.3.rs-3750712/v1

Additional Declarations: No competing interests reported.

Version of Record: A version of this preprint was published at World Wide Web on February 12th, 2024.
See the published version at https://doi.org/10.1007/s11280-024-01251-w.

LAMEE: A Light All-MLP Framework for Time Series Prediction Empowering Recommendations

Yi Xie[1,2], Yun Xiong[1,2], Xiaofeng Gao[3*], Jiadong Chen[3], Yao Zhang[1,2], Xian Wu[1,2], Chao Chen[4]

[1*]Shanghai Key Lab of Data Science, Fudan University, Shanghai, 200433, China.
[2]School of Computer Science, Fudan University, Shanghai, 200433, China.
[3]MoE Key Lab of Artificial Intelligence, Shanghai Jiaotong University, Shanghai, 200240, China.
[4]College of Computer Science, Chongqing University, Chongqing, 400044, China.

*Corresponding author(s). E-mail(s): gao-xf@cs.stju.edu.cn;
Contributing authors: yixie18@fudan.edu.cn; yunx@fudan.edu.cn; chenjiadong998@sjtu.edu.cn; yaozhang18@fudan.edu.cn; xianwu21@fudan.edu.cn; cschaochen@cqu.edu.cn;

Abstract

Exogenous variables, unrelated to the recommendation system itself, can significantly enhance its performance. Therefore, integrating these time-evolving exogenous variables into a time series and conducting time series predictions can maximize the potential of recommendation systems. We refer to this task as Time Series Prediction Empowering Recommendations (TSPER). However, as a subtask within the recommendation system, TSPER faces unique challenges such as computational and data constraints, system evolution, and the need for performance and interpretability. To meet these unique needs, we propose a lightweight Multi-Layer Perceptron architecture with joint Time-Frequency information, named **L**ight **A**ll-**M**LP with joint Tim**E**-fr**E**quency information (LAMEE). LAMEE utilizes a lightweight MLP architecture to achieve computing efficiency and adaptive online learning. Moreover, various strategies have been employed to improve the model, ensuring stable performance and model interpretability. Across multiple time series datasets potentially related to recommendation systems, LAMEE balances performance, efficiency, and interpretability, overall surpassing existing complex methods.

Keywords: time series analysis, multi-layer perceptron, joint time-frequency information, lightweight framework, recommendation systems

1 Introduction

Recommendation systems have become a pivotal component in various online platforms, offering numerous benefits and applications that enhance user experience and business performance [1]. Research has found that in some recommendation scenarios, considering some exogenous knowledge often helps to enhance the performance [2–4]. For instance, considering stock market information can aid in recommending financial products [5, 6]; taking into account illness information can strengthen recommendations in social media [7]; considering weather, traffic or some social information can assist in recommending POI or navigation strategies [8–11], *etc.* In general, exogenous knowledge can significantly affect the strategies of recommendations.

We refer to some numerical exogenous knowledge as exogenous variables, it can be therefore concluded that accurately predicting the future trends of these exogenous variables can effectively assist the recommendation system in enhancing its performance, both now and in the future. The prediction of exogenous variables is essentially a time series prediction and modeling problem. We refer to this kind of task as the Time Series Prediction Empowering Recommendations (TSPER) task. Compared to traditional time series forecasting and modeling tasks, TSPER is a subsystem embedded within the recommendation systems and therefore has some additional requirements:

- **Computing Efficiency:** TSPER task is designed to support the recommendation system, so it should not consume too many resources originally allocated to the recommendation system. It even needs to be efficient in inference and training phases in GPU-less environments, which requires lightweight frameworks.
- **Adaptive Online Learning:** TSPER must adapt to changes in the content of the recommendation system over time through online training, which means that this task must have the ability to tolerate a certain degree of 'distribution drift' to adapt to new recommendation patterns. This allows the model to quickly absorb new information and discard outdated data, in order to maintain its temporal relevance to the recommendation system.
- **Assured Performance:** Despite the above limitations, we still require this model to have excellent prediction performance, which is fundamental to a superior prediction model. Sacrificing the performance of the model to achieve efficiency and model lightweight is a case of missing the forest for the trees.
- **Model Interpretability:** As a time series prediction tool aiding the recommendation system, the model's interpretability is critical for business analysis, ensuring that its functionality and outputs are comprehensible and actionable.

To this end, we propose **L**ight **A**ll-**M**LP with joint Tim**E**-fr**E**quency information (LAMEE). As the name implies, it is an all-MLP framework leveraging information from both time and frequency domains.

To meet the requirements of **Computing Efficiency**, LAMEE has chosen Multi-Layer Perceptrons (MLP) as the fundamental building blocks for its time series model. Compared to models based on Transformer [12–16], the lightweight and efficient nature of MLPs gives them a computational efficiency edge. Particularly in GPU-less environments, LAMEE demonstrates impressive training speeds on CPUs, even surpassing some Transformer and convolution-based models in their GPU training speeds.

For **Adaptive Online Learning**, LAMEE's MLP components are effective. Recommendation system content changes over time, which can be regarded as a phenomenon of "concept drift" [17, 18]. While MLPs are more sensitive to these shifts compared to Transformers [19, 20], this sensitivity is advantageous for the TSPER task. Due to the need for timely recommendations, limited training data, and the MLP's efficiency in training, its ability to quickly adapt to new data is beneficial. Although MLPs might not capture long-term dependencies as well as Transformer-based models, their rapid adaptability is crucial for recommendation systems that require fast responses.

Considering these constraints, LAMEE's performance can be affected. For **Assured Performance**, we propose three strategies to counter that: joint time-frequency modeling, residual decomposition, and input-auxiliary supervision strategy. These strategies, essentially, can be regarded as feature augmentation, high-frequency components denoise, and trend correction, respectively, which significantly boost LAMEE's prediction performance. Additionally, LAMEE prefers raw features over high-dimensional embeddings, avoiding unnecessary information redundancy caused by the non-full rank nature of high-dimensional spaces [21–23], while maintaining **Model Interpretability**. Furthermore, leveraging the denoising capability of frequency information via low-pass filtering [24, 25], and its comprehensive perception across the entire time domain, we devise residual decomposition and input-auxiliary supervision strategies. These strategies fortify noise immunity and tackle the issue of inconsistent distribution between input signals and predictions (termed "trend inconsistent" in this context), substantially improving performance.

Extensive experiments demonstrate, on six time-series datasets related to recommendation systems, LAMEE achieved the top two positions in 45 out of 48 metrics, among which, in the 34 metrics where LAMEE had the best performance, the average improvement over the second place was 16.7%. Based on the above, efficiency discussions indicate that LAMEE, compared to these baselines, only used an average of 29.48% of the parameter amount, 12.07% of the time consumption, and 19.84% of the memory usage. Further experiments also demonstrated that LAMEE possesses excellent adaptive online learning capabilities and interpretability.

The primary contributions of this paper are outlined as follows:

- The paper emphasizes the importance of time series data in capturing changes in user behavior and preferences over time in recommendation systems, and highlights the challenges in processing such data. We refer to such task as Time Series Prediction Empowering Recommendations (TSPER) task.
- We propose **L**ight **A**ll-**M**LP with joint tim**E**-fr**E**quency information (LAMEE) to handle these challenges. The LAMEE model employs an all-Multilayer Perceptron (MLP) architecture and integrates time and frequency information, aiming to

enhance the handling of time series data. It addresses specific challenges in recommendation systems through strategies focused on computational efficiency, adaptive online learning, performance assurance, and interpretability.
- Through a series of experiments, the LAMEE model demonstrates its superior performance in various recommendation scenarios, particularly in handling concept drift and maintaining computational efficiency, showing significant advantages over other models.

2 Related Works

2.1 Time Series Prediction

Time series prediction remains an essential tool in various domains, from finance and healthcare to energy forecasting and stock predictions. Over the decades, the methods for time series prediction have evolved significantly.

Historically, statistical methods such as autoregressive moving average models [26, 27], exponential smoothing [28], and Kalman filters [29] were dominant. These methods, while powerful, often required domain-specific knowledge and manual calibration for optimal performance. Machine learning ushered in a new era with techniques like gradient boosting regression tree (GBDT) [30, 31]. While they outperformed many traditional methods, they often depended heavily on manual feature extraction and engineering to be effective.

The rise of deep learning brought a paradigm shift. Neural networks, with their ability to automatically learn representations from raw data, presented a natural fit for time series prediction. Initial forays into this area saw the deployment of Recurrent Neural Networks (RNNs) [32–34], which were specifically designed to handle sequences. Temporal Convolutional Neural Networks (TCNs) [35–37] soon followed, leveraging convolutional layers to capture local and global patterns in time series data.

Transformers and their variants have also emerged as a popular approach for time series prediction, given their self-attention mechanism's ability to capture long-range dependencies [12]. However, as noted, the quadratic complexity in attention operations can be computationally intensive, especially for longer sequences. Alternative architectures that strike a balance between computational efficiency and predictive power are still a topic of active research.

2.2 The Renaissance of MLPs

Multi-Layer Perceptrons (MLPs), often considered the building blocks of neural networks, have enjoyed a resurgence in recent years. Their foundational work dates back to the early 1990s [38–40]. While initially overshadowed by specialized architectures like CNNs and RNNs, MLPs have remained the workhorse for various tasks, forming the backbone of architectures such as CNNs [41], LSTM networks [42], and even Transformers [12].

The recent success of Transformers in diverse fields like computer vision [43], language processing [44], data mining [45], and time series prediction [46], has inadvertently shone a spotlight back on MLPs. Some research argues that, given the right

conditions, MLPs can outperform more sophisticated models in tasks as diverse as image classification, language understanding, and more [47, 48].

It's posited that the architectural simplicity of an MLP, when scaled appropriately, can often lead to better generalization and easier optimization [**?**]. This resurgence has spurred discussions on understanding the underlying reasons behind the efficacy of MLPs. Some believe that it offers an architectural inductive bias that aids in particular tasks. Recent studies [49] have even sought to uncover the relationship between MLPs and their more intricate counterparts like CNNs.

3 Problem Formulation

We first formulate the problem as follows. Let $\mathbf{X}(t) \in \mathbb{R}^c$ and $\mathbf{Y}(t) \in \mathbb{R}^c$ denote the real values, and the predicted values at the timestamp t, c is the channels (or the number of variables). Specifically, we use $\mathbf{X}_k(t), \mathbf{Y}_k(t) \in \mathbb{R}$ to denote the value of the k-th channel at timestamp t. Given a set of I-length ordered history information $\mathbf{X}(0:I-1) \in \mathbb{R}^{I\times c} = [\mathbf{X}(0), \mathbf{X}(1), \cdots, \mathbf{X}(I-1)]$ from timestamp 0 to timestamp $I-1$ as the input signals, the target is to predict the next O-timestamp values: $\mathbf{Y}(I:I+O-1) \in \mathbb{R}^{O\times c} = [\mathbf{Y}(I), \mathbf{Y}(I+1), \cdots, \mathbf{Y}(I+O-1)]$, from the timestamp I to the timestamp $I+O-1$. Analogously, $\mathbf{X}_k(0:I-1) \in \mathbb{R}^I$ and $\mathbf{Y}_k(I:I+O-1) \in \mathbb{R}^O$ respectively denote the iuput and predicted sequences of the corresponding channel. In the follows, $\mathbf{X}(0:I-1)$ and $\mathbf{Y}(I:I+O-1)$ will be simplified denoted as $\mathbf{X}$ and $\mathbf{Y}$.

4 Methodology

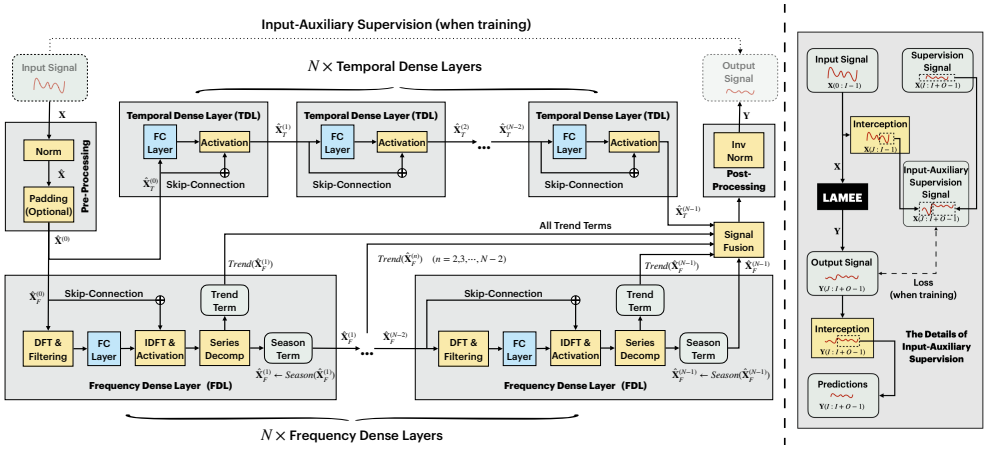

Fig. 1 (Left) Depicting LAMEE's overall architecture. (Right) A detailed view of the input-auxiliary supervision strategy. The LAMEE black box corresponds to the entirety of the structure on the left.

4.1 Overview

Figure 1 presents the simplified, lightweight architecture of LAMEE. The input signals are denoted as $\mathbf{X} \in \mathbb{R}^{I \times c}$, and $\mathbf{X}_k \in \mathbb{R}^I$ denotes the k-th channel of input signals.

4.1.1 Pre-Processing

The aim of pre-processing in LAMEE is to transform raw input signals $\mathbf{X}$ into the desired format $\hat{\mathbf{X}}$. This involves a necessary normalization step and an optional padding operation.

Normalization (Norm). For the stability of the input signals distributions during training, we will first perform time-wise normalization for the input signals to ensure the distributions of the input signals are within 0 and 1:

$$Norm(\mathbf{X}_k) = \frac{\mathbf{X}_k - mean(\mathbf{X}_k)}{std(\mathbf{X}_k)}. \tag{1}$$

Padding (Optional). As previously stated, we employ an input-auxiliary supervision strategy to mitigate the issue of trend drift. Consequently, the output signals have a length of $(J + O)$, where J is a hyper-parameter associated with this strategy and $J \leq I$ (refer to Section 4.4 for more details). An optional padding operation can be utilized to bring input signals to the desired length:

$$\hat{\mathbf{X}}^{(0)} = concat(\mathbf{X}, \mathbf{Z}) \tag{2}$$

where $\mathbf{Z} \in \{0\}^{J \times c}$ is an all-zero tensor, $\hat{\mathbf{X}}^{(0)} \in \mathbb{R}^{(J+O) \times c}$. Subsequent operations will be performed on the full $(J+O)$-length signals in both the time and frequency domains. Here, we use a superscript to denote the layer index of the output. For example, $\hat{\mathbf{X}}^{(0)}$ signifies the output of the 0-th layer,*i.e.*, the input to the whole model. It should be noted that the padding operation's goal is to maintain the consistency of signal shapes across layers, though this isn't strictly necessary.

4.1.2 Post-Processing

The aim of the post-processing step is to transform the output signals $\hat{\mathbf{Y}}$ into a form $\mathbf{Y}$ with tangible significance. This involves a single operation: inverse normalization.

Inverse Normalization (InvNorm). Since the distribution of output signals falls within the range of 0 to 1, we carry out a time-wise inverse normalization to ensure that the final predictions align with actual distributions:

$$InvNorm(\mathbf{Y}_k) = \mathbf{Y}_k \odot std(\mathbf{X}_k) + mean(\mathbf{X}_k). \tag{3}$$

Here, $mean(\cdot)$ and $std(\cdot)$ represent time-wise mean values and standard deviations, respectively, while $\odot$ signifies an element-wise product.

4.2 Time & Frequency Dense Layer

Once pre-processed, the resulting input signals $\hat{\mathbf{X}}^{(0)}$ will be divided into two branches, which are sent into the temporal dense layer and the frequency dense layer, denoted as $\hat{\mathbf{X}}_T^{(0)}$ and $\hat{\mathbf{X}}_F^{(0)}$ respectively. The objective is to perform non-linear transformations, capturing information that is hard to acquire solely within a single domain.

4.2.1 Temporal Dense Layer

Temporal Dense Layers (TDLs) are to execute non-linear transformations within the time domain, comprising a fully-connected layer (FC Layer), an activation function, and a skip-connection. The operations within each layer can be articulated as follows:

$$\hat{\mathbf{X}}_T^{(n+1)} = TDL(\hat{\mathbf{X}}_T^{(n)}) = GELU(\mathcal{W}_1^{(n)}\hat{\mathbf{X}}_T^{(n)}\mathcal{W}_2^{(n)}) + \hat{\mathbf{X}}_T^{(n)}. \tag{4}$$

At the n-th temporal dense layer, $\mathcal{W}_1^{(n)} \in \mathbb{R}^{(J+O)\times(J+O)}$ and $\mathcal{W}_2^{(n)} \in \mathbb{R}^{c\times c}$ are trainable parameters, where $\mathcal{W}_1^{(n)}$ is the time-wise transformation parameter matrix, and $\mathcal{W}_2^{(n)}$ is for channel-wise transformations. $GELU(\cdot)$ denotes the Gaussian Error Linear Units activation function [50].

4.2.2 Frequency Dense Layer

The Frequency Dense Layers (FDLs) are employed for non-linear transformations in the frequency domain. For the n-th layer input signals $\hat{\mathbf{X}}_F^{(n)} \in \mathbb{R}^{(J+O)\times c}$, the FDL processing pipeline is as follows:

- Discrete Fourier Transforms (DFT):

$$\mathcal{F}^{(n)}(\omega) = DFT(\hat{\mathbf{X}}_F^{(n)})(\omega) = \sum_{t=0}^{(J+O)-1} \hat{\mathbf{X}}_F^{(n)}(t) e^{-\frac{2\pi\omega t}{J+O} i}, \tag{5}$$

  where $\mathcal{F}^{(n)} \in \mathbb{C}^{(\lfloor \frac{(J+O)}{2} \rfloor + 1)\times c}$ denotes the bases in the frequency domain, containing $\lfloor \frac{(J+O)}{2} \rfloor + 1$ frequency components. ω is a frequency-correlated variable [1].
- Low-Pass Filtering (Filtering). Selecting the lowest $M \leq \lfloor \frac{(J+O)}{2} \rfloor + 1$ frequency components as the new bases, and other frequency components are ignored. This process helps identify noise while reducing computational complexity. Afterwards, $\mathcal{F}^{(n)} \in \mathbb{C}^{M\times c}$.
- Fully-Connected Layer (FC Layer):

$$\mathcal{F}^{(n)} \leftarrow \mathcal{U}_1^{(n)}\mathcal{F}^{(n)}\mathcal{U}_2^{(n)}, \tag{6}$$

  $\mathcal{U}_1^{(n)} \in \mathbb{C}^{M\times M}$ and $\mathcal{U}_2^{(n)} \in \mathbb{C}^{c\times c}$ denote the complex trainable parameters at the n-th frequency dense layer for frequency-wise and channel-wise transformation, respectively.
- Inverse Discrete Fourier Transforms (IDFT):

$$IDFT(\mathcal{F}^{(n)}) = \frac{1}{(J+O)} \sum_{\omega=0}^{(J+O)-1} \mathcal{F}^{(n)}(\omega) e^{\frac{2\pi\omega t}{J+O} i}. \tag{7}$$

- Activation and skip connection. In FDL, we still use Gaussian Error Linear Units as the activation function, and also add the skip connection:

$$\hat{\mathbf{X}}_F^{(n+1)} = FDL(\hat{\mathbf{X}}_F^{(n)}) = GELU(IDFT(\mathcal{F}^{(n)})) + \hat{\mathbf{X}}_F^{(n)}. \tag{8}$$

The output from the temporal dense layers feeds into the subsequent temporal dense layer, while the output from the frequency dense layers undergoes residual decomposition.

---

[1] Here, we use single side spectrum for simplification, resulting in there are $\lfloor \frac{(J+O)}{2} \rfloor + 1$ frequency components, rather than $(J+O)$.

4.3 Residual Decomposition

Low-pass filtering in signal processing identifies noise usually found in the highest frequency components. But, because of potential overlap between noise and high-frequency data, simple filtering could lose important data or retain harmful noise. Transformers can partly manage noise through their attention mechanism, the simpler MLPs are more susceptible. Small amounts of noise can greatly impact prediction performance, showing MLPs' reduced noise tolerance, a significant concern given the ubiquity of real-world noise.

To combat noise interference, we introduce a residual decomposition method. Based on time series theories, we split a time series into trend and seasonal elements after each Frequency Dense Layer. These correspond to the low and high-frequency components of the series, respectively. Each seasonal element is processed in the next Frequency Dense Layer, undergoing low-pass filtering, feature transformation, and further decomposition. Meanwhile, each trend element directly feeds into the final fusion and prediction process.

Residual decomposition operates on the premise that noise usually exists in signals' highest frequency components [51]. It distinguishes between high-frequency components and noise, recognizing that higher frequencies likely indicate noise, while lower frequencies carry genuine information. By consistently extracting and filtering out higher frequency components, it reduces noise while preserving crucial high-frequency data. This approach minimizes the risk of mishandling these components and noise, enhancing the accuracy of noise detection and the model's noise filtration capabilities. Essentially, residual decomposition combines prior knowledge from series decomposition with N-Beats' residual learning.

4.3.1 Decomposition Strategy

We employ an additive decomposition model [52] to decompose signals into trend and seasonal terms, which has been widely adopted in various works [15, 53].

$$\begin{aligned} Trend(\hat{\mathbf{X}}_F^{(n)}) &= AvgPool(\hat{\mathbf{X}}_F^{(n)}) \\ Season(\hat{\mathbf{X}}_F^{(n)}) &= \hat{\mathbf{X}}_F^{(n)} - Trend(\hat{\mathbf{X}}_F^{(n)}) \end{aligned}. \tag{9}$$

4.3.2 Signal Fusion

We fuse all trend terms, the output of TDL and FDL as the final output signals. In this paper, we use the simplest additive model to fuse all trend terms and the output of temporal and frequency dense layers. Suppose we have N temporal and frequency dense layers, the final output signals $\mathbf{Y}$ can be represented as:

$$\begin{aligned} \mathbf{Y} &= \sum_{n=0}^{N-1} Trend(\hat{\mathbf{X}}_F^{(n)}) + \hat{\mathbf{X}}_F^{(N-1)} + \hat{\mathbf{X}}_T^{(N-1)} \\ where \quad \hat{\mathbf{X}}_F^{(n+1)} &= FDL(Season(\hat{\mathbf{X}}_F^{(n)})) \quad if \quad n \neq 0 \end{aligned} \tag{10}$$

in which, $\mathbf{Y} \in \mathbb{R}^{(J+O)\times c}$.

4.4 Input-Auxiliary Supervision

Applying MLPs directly to time series prediction can lead to a situation where the predictions and input signals differ in numerical distribution, even to the point of magnitude discrepancies. We term this situation as "trend inconsistent". Due to the auto-regressive nature of time series, there should be a high correlation between input signal and prediction. This makes the trend inconsistent phenomenon detrimental to time series prediction tasks, and it becomes particularly prominent in non-stationary time series. This problem is primarily due to the inherent limitations in the representational capacity of MLPs.

To address this problem, we introduce an optimization strategy termed "input-auxiliary supervision". The central concept of input-auxiliary supervision involves having part or all of the input signals contribute to the model's supervision and optimization along with the original supervision signals. This concept, akin to AutoEncoders, compels the model to reconstruct part or all of the input signals. Given the high correlation between input and prediction in time series, such a reconstruction forces the numerical distributions of predictions to align more closely with the input signals in the time domain, thus mitigating trend inconsisten. Furthermore, in the frequency domain, as frequency information provides a "global perception" across the entire time domain, reconstructing the input signals also rectifies the final predictions.

More specifically, for input signals of length I, we reconstruct the most recent J-length signals. The reconstructed signals are then outputted together with the predictions, resulting in a combined output signal of length $(J + O)$. During training, we incorporate the most recent J-length input signals $\mathbf{X}(J : I - 1)$ to form the input-auxiliary supervision signals $\mathbf{X}(J : I + O - 1)$ by concatenating them with the supervision signals (ground truth) $\mathbf{X}(I : I+O-1)$. These supervision signals are then used to optimize the model using mean squared errors (MSE) as the loss function:

$$Loss = MSE(\mathbf{Y}(J : I + O - 1), concat(\mathbf{X}(J : I - 1), \mathbf{X}(I : I + O - 1))). \tag{11}$$

During evaluation, we extract the prediction portion of the output signals $Y(I : I+O-1)$ to calculate metrics against the ground truth $X(I : I+O-1)$. If we regard the entirety of LAMEE as a black box, the input-auxiliary supervision strategy can be visually represented by the right side of Figure 1.

4.5 Discussion

4.5.1 Representation Learning

Numerous deep learning methods typically project representations into a high-dimensional space to create high-dimensional embeddings, the dimensions of which are independent of the original feature dimension. Such high-dimensional embeddings can markedly improve the representational capacity of deep learning methods. However, we have empirically found that high-dimensional embeddings can degrade performance in LAMEE, with detailed results provided in Section 5.7.

From the perspectives of linear spaces and tensor analysis, we conjecture that high-dimensional embeddings are equivalent to projecting low-dimensional raw features into a higher-dimensional space. Given that the raw features of time series are

Table 1 The theoretical comparison about efficiency and complexity. L denotes the length of time series, d represents the dimensions of latent representations, M is a hyper-parameter that determines the passed frequency components after filtering, and c denotes the channels. Usually, $M \leq \frac{L}{2}$, and $c < d$.

Methods	Complexity Function	Number of Operations	Parameter Amounts
Transformer [12]	$4Ld^2 + 2L^2d$	7	$4d^2$
Informer [13]	$4Ld^2 + 2(L \cdot \log L)d$	8	$4d^2$
Autoformer [14]	$4Ld^2 + 4(L \cdot \log L)d$	10	$4d^2$
Fedformer [15]	$4Ld^2 + 2M^2d + 2(L \cdot \log L)d$	8	$4d^2$
LAMEE (ours)	$2Lc^2 + L^2c + M^2c + 2(L \cdot \log L)c$	6	$L^2 + M^2 + 2c^2$

full rank, their high-dimensional projection inevitably results in a non-full rank, leading to additional information redundancy. Handling such high-dimensional redundant information requires powerful deep learning models, such as the Transformer and its variants. Yet, as the fundamental building block of LAMEE, the simple architecture of MLPs cannot effectively handle high-dimensional non-full rank features, leading to overfitting. Thus, we choose to use raw features directly instead of high-dimensional embeddings, thereby avoiding such additional complexity.

4.5.2 Efficiency and Complexity

We perform a simple theoretical analysis comparing the efficiency and complexity of LAMEE with several other baseline methods in terms of time complexity and parameters. The baseline methods included in this comparison are the Transformer, Informer, Autoformer, and Fedformer. We provide a comparison in terms of the theoretical complexity function, the number of operations, and the quantity of parameters for each layer. The results of this comparison can be found in Table 1.

The efficiency and complexity benefits of LAMEE primarily arise from two factors: (1) the number of channels, c, in raw features is considerably less than the dimension, d, of the high-dimensional latent space, and (2) fewer operations are performed. The threshold for the complexity function of LAMEE being less than all baselines is set at $Lc < d^2$. Here, we perform a simple deduction.

Proof. Let's consider the conditions $c < d$ and $M < L$, we then explore when the complexity function of LAMEE is smaller than that of Informer. This is given by the inequality

$$L^2c + M^2c + 2(L \cdot \log L)c + 2Lc^2 < 4Ld^2 + 2(L \cdot \log L)d. \tag{12}$$

Given the condition $c < d$, it follows that

$$2(L \cdot \log L)c < 2(L \cdot \log L)d, \tag{13}$$

and

$$2Lc^2 < 2Ld^2. \tag{14}$$

Thus, we can deduce:

Table 2 The statistics of datasets.

Datasets	Number of Variables (channels)	Sampling Frequency	Total Observastions	Splitting (train:val:test)
ILI	7	1 Week	966	7:1:2
Stock	654	1 Day	1681	7:1:2
Electricity	321	1 Hour	26304	7:1:2
Traffic	862	1 Hour	17544	7:1:2
Weather	21	10 Minutes	52695	7:1:2
ETTm2	7	15 Minutes	69680	6:2:2

$$L^2c + M^2c < 2Ld^2. \tag{15}$$

Now consider the condition $M < L$, this implies that

$$L^2c + M^2c \leq 2L^2c < 2Ld^2, \tag{16}$$

From this inequality, we conclude:

$$Lc < d^2. \tag{17}$$

Hence, when $c < d$ and $M < L$ hold, a sufficient but not necessary condition for the complexity function of LAMEE to be smaller than that of Informer is $Lc < d^2$. □

Further experiments addressing efficiency and complexity are also conducted in Section 5.3.

5 Experiments

5.1 Experimental Settings

5.1.1 Dataset

We conduct experiments on six datasets with varying domains: **Influenza-Like Illness (ILI)**[1], **Stock**[2], **Electricity (ELC)**, **Traffic**[3], **Weather**, **Electricity Transformer Temperature (ETTm2)**[4]. The detailed information and statistics are summarized in Table 2.

We simply categorize these datasets as three kinds:

- Small Datasets. We classify the Influenza-Like Illness (ILI) and Stock datasets as small due to their limited size. These datasets help assess the model's ability to represent and generalize from a small amount of data.
- Multi-Variable Datasets. Electricity and Traffic are multi-variable datasets. The commonality of both is: each channel has the same physical meaning, thus the evolution and distributions among channels are overall consistent.
- Multi-Source Datasets. Weather and ETTm2 datasets are multi-source datasets. Each channel of these datasets has completely different physical meanings. This

[1]https://gis.cdc.gov/grasp/fluview/fluportaldashboard.html
[2]https://www.csmar.com/channels/31.html
[3]http://pems.dot.ca.gov/
[4]https://github.com/zhouhaoyi/ETDataset

leads to inconsistent evolution and distributions among chennels, which increases the difficulty of prediction.

For ILI dataset, we take the length of input as $I = 36$, and output $O \in \{24, 36, 48, 60\}$-length predictions. For Stock dataset, we take the length of input as $I = 48$, and output $O \in \{12, 24, 48, 96\}$-length predictions. For other datasets, we take the length of input as $I = 96$, and output $O \in \{96, 192, 336, 720\}$-length predictions. These differences are mainly due to the different amounts of data.

5.1.2 Baselines and Metrics

We compared LAMEE with the following 13 baselines: **MTGNN** [54], **Informer** [13], **Autoformer** [14], **Pyraformer** [16], **Fedformer** [15], **Non-Stationary Transformer (NSTrans)** [55], **ETSformer** [56], **MICN** [57], **DLinear** [53], **LightTS** [58], **CrossFormer** [59], **TimesNet** [60]. We use Mean Absolute Errors (MAE) and Mean Sequared Errors (MSE) as the metrics, widely used in previous studies.

5.2 Main Results

The main comparison results are given in Table 3. In a macro view, LAMEE achieves state-of-the-art performance on 38 out of 48 metrics in six datasets, and achieves the top two performance on 46 out of which. On the all metrics that LAMEE achieves the best, the averaged improvement is 11.09%.

5.2.1 Small Datasets

LAMEE outperforms in all metrics for the ILI and Stock datasets, averaging improvements of 34.20% and 17.03% respectively. These results demonstrate LAMEE's excellent prediction abilities on small datasets. Usually, small datasets often complicate model training due to their insufficient sample size. Conversely, LAMEE's light all-MLP architecture and direct use of raw features, as opposed to high-dimensional embeddings, allows for better fitting and more efficient training on small datasets.

5.2.2 Multi-Variable Datasets

While LAMEE doesn't top all metrics on the Electricity and Traffic datasets, it still surpasses all baselines on 6 out of 16 metrics (with averaged 2.77% performance improvements) and achieves top-two performance in 14 of them. The relative consistency in the evolution and distribution between these two datasets, as well as the clear seasonal patterns, contribute to the existing methods performing well on these datasets, which results in LAMEE not exhibiting a significant advantage over them. Nonetheless, LAMEE has achieved excellent performance.

5.2.3 Multi-Source Datasets

LAMEE scores average improvements of 6.98% and 5.58% on all metrics for the Weather and ETTm2 datasets respectively. These multi-source datasets each consist of channels with distinct physical meanings, making prediction more challenging

Table 3 Main Results. The prediction errors in terms of MSE and MAE, the lower the better. **Bold fonts** denote the best performance, and underline denotes the second best.

Datasets	Steps	Metrics	MTGNN	Informer	Autoformer	Pyraformer	Fedformer	NSTrans	ETSformer	MICN	DLinear	LightTS	CrossFormer	TimesNet	**LAMEE**
ILI	24	MSE	4.265	5.764	3.483	7.394	3.228	2.294	2.527	2.684	2.248	8.313	3.041	2.317	**0.832**
		MAE	1.387	1.677	1.287	2.012	1.260	0.945	1.020	1.112	1.011	2.144	1.186	0.934	**0.588**
	36	MSE	4.777	4.755	3.103	7.551	2.679	1.825	2.615	2.507	2.436	6.631	3.406	1.972	**1.083**
		MAE	1.496	1.467	1.148	2.031	1.080	0.848	1.007	1.013	1.019	1.902	1.232	0.920	**0.643**
	48	MSE	5.333	4.763	2.669	7.662	2.622	2.010	2.359	2.423	2.414	7.299	3.459	2.238	**1.157**
		MAE	1.592	1.469	1.085	2.057	1.078	0.900	0.972	1.012	1.051	1.982	1.221	0.940	**0.707**
	60	MSE	5.070	5.264	2.770	7.931	2.857	2.178	2.487	2.653	2.514	7.283	3.640	2.027	**1.469**
		MAE	1.552	1.564	1.125	2.100	1.157	0.963	1.016	1.085	1.116	1.985	1.305	0.928	**0.767**
Stock	12	MSE	3.772	5.362	3.911	5.264	3.852	4.041	4.453	4.133	3.792	5.508	4.193	3.990	**2.592**
		MAE	1.309	1.244	1.153	1.271	1.119	1.107	1.246	1.096	0.989	1.621	1.437	1.030	**0.718**
	24	MSE	4.892	5.859	4.196	5.486	4.123	4.457	4.470	4.241	3.888	5.103	4.021	4.512	**3.123**
		MAE	1.330	1.307	1.178	1.484	1.142	1.131	1.393	1.090	1.021	1.467	1.220	1.092	**0.814**
	48	MSE	4.613	5.965	4.620	5.931	4.553	4.139	4.933	4.466	4.513	5.321	4.480	4.335	**3.774**
		MAE	1.588	1.308	1.228	1.520	1.191	1.306	1.517	1.293	1.322	1.461	1.291	1.096	**0.920**
	96	MSE	5.541	5.986	5.345	5.998	5.335	5.505	5.784	5.167	5.013	5.593	5.463	5.331	**4.595**
		MAE	1.507	1.310	1.300	1.573	1.295	1.241	1.513	1.181	1.231	1.668	1.420	1.253	**1.039**
ELC	96	MSE	0.272	0.274	0.201	0.386	0.193	0.169	0.187	0.165	0.176	0.207	0.171	0.168	**0.158**
		MAE	0.361	0.368	0.317	0.449	0.308	0.273	0.304	0.276	**0.252**	0.307	0.273	0.272	0.264
	192	MSE	0.297	0.296	0.222	0.378	0.201	0.182	0.199	0.187	0.181	0.213	0.196	0.184	**0.176**
		MAE	0.380	0.386	0.334	0.443	0.315	0.286	0.315	0.296	0.285	0.316	0.287	0.289	**0.277**
	336	MSE	0.327	0.296	0.222	0.378	0.201	0.182	0.199	0.193	0.198	0.213	0.205	0.198	**0.188**
		MAE	0.383	0.394	0.338	0.443	0.329	0.304	0.329	0.298	0.296	0.333	0.317	0.300	**0.290**
	720	MSE	0.420	0.373	0.254	0.376	0.246	0.222	0.233	**0.207**	0.245	0.265	0.241	0.220	0.237
		MAE	0.410	0.439	0.361	0.445	0.355	0.321	0.345	0.316	0.333	0.360	0.353	0.320	**0.310**
Traffic	96	MSE	0.651	0.719	0.613	0.867	0.587	0.612	0.607	0.602	0.611	0.615	**0.525**	0.593	0.567
		MAE	0.413	0.391	0.388	0.468	0.366	0.338	0.392	0.373	0.391	0.391	**0.296**	0.321	0.328
	192	MSE	0.682	0.696	0.616	0.869	0.604	0.613	0.621	0.612	0.587	0.601	**0.529**	0.617	0.562
		MAE	0.404	0.379	0.382	0.467	0.373	0.340	0.399	0.402	0.351	0.382	**0.297**	0.336	0.317
	336	MSE	0.700	0.777	0.622	0.881	0.621	0.618	0.622	0.637	0.610	0.613	**0.530**	0.629	0.597
		MAE	0.416	0.420	0.337	0.469	0.383	0.328	0.396	0.439	0.393	0.386	**0.300**	0.336	0.330
	720	MSE	0.741	0.864	0.660	0.896	0.626	0.653	0.632	0.639	0.645	0.658	**0.573**	0.640	0.631
		MAE	0.435	0.472	0.408	0.473	0.382	0.355	0.396	0.413	0.394	0.407	**0.313**	0.350	0.348
Weather	96	MSE	0.408	0.300	0.266	0.622	0.217	0.173	0.197	0.192	0.172	0.182	0.248	0.172	**0.153**
		MAE	0.441	0.384	0.336	0.556	0.296	0.223	0.281	0.250	0.255	0.242	0.318	0.220	**0.199**
	192	MSE	0.452	0.598	0.307	0.739	0.276	0.245	0.237	0.240	0.237	0.227	0.251	0.219	**0.203**
		MAE	0.513	0.544	0.367	0.624	0.336	0.285	0.312	0.300	0.296	0.287	0.369	0.261	**0.248**
	336	MSE	0.668	0.578	0.359	1.004	0.339	0.321	0.298	0.281	0.283	0.282	0.335	0.280	**0.258**
		MAE	0.805	0.523	0.395	0.753	0.380	0.338	0.353	0.330	0.335	0.334	0.415	0.306	**0.291**
	720	MSE	0.940	1.059	0.419	1.420	0.403	0.414	0.352	0.350	0.345	0.352	0.420	0.365	**0.330**
		MAE	1.039	0.741	0.428	0.934	0.428	0.410	0.388	0.387	0.381	0.386	0.499	0.359	**0.340**
ETTm2	96	MSE	0.688	0.365	0.255	0.435	0.203	0.192	0.189	0.190	0.193	0.209	0.273	0.187	**0.177**
		MAE	0.602	0.453	0.339	0.507	0.287	0.274	0.280	0.285	0.292	0.308	0.356	0.267	**0.255**
	192	MSE	0.851	0.533	0.281	0.730	0.269	0.280	0.253	0.284	0.284	0.311	0.426	0.249	**0.241**
		MAE	0.732	0.563	0.340	0.673	0.328	0.339	0.319	0.356	0.362	0.382	0.487	0.309	**0.296**
	336	MSE	0.941	1.363	0.339	1.201	0.325	0.334	0.314	0.394	0.369	0.442	0.516	0.321	**0.297**
		MAE	0.796	0.887	0.372	0.845	0.366	0.361	0.357	0.430	0.427	0.466	0.631	0.351	**0.330**
	720	MSE	1.401	3.379	0.433	3.625	0.421	0.417	0.414	0.537	0.554	0.675	0.592	0.408	**0.397**
		MAE	1.079	1.338	0.432	1.451	0.415	0.413	0.413	0.509	0.522	0.587	0.673	0.403	**0.390**

due to unique numerical distributions and evolution patterns. Yet, LAMEE performs well by efficiently modeling these patterns with raw features, avoiding non-full rank redundancy from high-dimensional spaces. Its input-auxiliary supervision also reduces numerical biases by mitigating trend inconsistent, and joint time-frequency modeling further improves performance.

5.3 Efficiency Discussion

In a previous section, we compared LAMEE with other methods by looking at complexity, operations, and parameters in Section 4.5.2. Now, we test these findings with actual experiments, focusing on parameters, time consumption (on both GPUs and CPUs), and memory usage.

We'll use both long and short-term scenarios on Electricity and Weather datasets to show LAMEE's effectiveness. In the long-term scenario, we use 1000 input signals to predict 1000 future timestamps, while in the short-term, both input and output are 100. We're comparing LAMEE with Informer [13], Crossformer [59], Autoformer [14], Fedformer [15], DLinear [53], MICN [57], and TimesNet [60], all using their settings at the best performance.

Table 4 Efficiency comparison. In the table, # Para denotes the number of parameters, TC denotes the time consumption (seconds per epoch), and MU denotes the memory usage on GPUs. OOM means out of memory.

Datasets		ELC				Weather			
Metrics		# Para	TC	TC on CPUs	MU	# Para	TC	TC on CPUs	MU
Methods	I, O								
Informer	100	12.88M	32.16	> 1000	2.57GB	11.51M	54.98	> 1000	2.43GB
	1000	24.74M	159.81	-	11.45GB	23.38M	197.30	-	7.98GB
Autoformer	100	12.57M	40.24	> 1000	3.48GB	10.73M	68.95	> 1000	3.41GB
	1000	28.46M	255.65	-	21.82GB	22.61M	430.91	-	19.74GB
Fedformer	100	17.97M	108.97	> 1000	3.01GB	16.44M	121.30	> 1000	2.83GB
	1000	22.84M	408.53	-	6.79GB	20.89M	603.24	-	3.41GB
DLinear	100	**0.11M**	18.65	56.51	1.09GB	**0.11M**	14.34	33.51	**0.91GB**
	1000	**11.26M**	100.26	-	3.71GB	**11.26M**	36.46	-	**1.77GB**
MICN	100	6.83M	24.31	325.35	2.15GB	1.59M	30.80	214.23	1.88GB
	1000	24.87M	108.63	-	5.19GB	21.89M	99.41	-	11.62GB
TimesNet	100	150.33M	590.41	> 1000	14.01GB	6.70M	106.35	> 1000	13.41GB
	1000	152.28M	> 1000	-	23.51GB	22.31M	>1000	-	21.48GB
LAMEE	100	0.67M	**10.32**	**46.36**	**0.89GB**	0.26M	**13.41**	**20.43**	1.03GB
	1000	20.44M	**87.43**	-	**3.51GB**	18.91M	**26.49**	-	1.89GB

5.3.1 Parameter Amounts

Our experiments show that LAMEE uses 0.44%-16.31% and 13.90%-90.73% of non-linear methods' parameters (excluding DLinear) in short-term and long-term cases, respectively. On average, it uses 37.1% of the parameters compared to other non-linear models in all cases. As signal length increases, LAMEE's parameter advantage diminishes due to its quadratic parameter usage. However, it still has a significant advantage in short-term cases in both datasets. While LAMEE lacks a parameter advantage against DLinear because of their internal structures, it still uses significantly fewer parameters than all other baselines

5.3.2 Time Consumption

On GPUs, LAMEE takes only 1.70%-44.21% and 4.4%-80.15% of the time that these non linear methods (excluding DLinear) use in short- and long-term tests, respectively. In all the cases listed, it takes averaging 21.9% of the time used by other non linear baselines. Even when considering a linear method such as DLinear, LAMEE demonstrates excellent efficiency. On CPUs, the efficiency of linear models like DLinear and LAMEE is even more evident, since only the two can be trained within 1000 seconds for an epoch, and other methods take over 1000 seconds. This makes them suitable for training in environments with computationally constrained environments (GPU-less). In general, LAMEE has great time efficiency in both short- and long-term cases on both GPUs and CPUs.

5.3.3 Memory Usage

LAMEE uses only 6.3%-54.81% and 8.8%-67.60% of the memory these non linear methods (excluding DLinear) take in short- and long-term cases, respectively. In all the cases listed, it takes averaging 26.7% of the memory used by other non linear baselines. Memory usage often matches the number of operations and complexity. Theoretically, LAMEE has fewer operations per layer and lower complexity than these models.

The efficiency metrics show that linear models: LAMEE and DLinear, perform better than Transformer and convolutional methods. Though similar, LAMEE excels by using fewer resources and less time. Despite potential variations, actual results do give insight into real-world efficiency.

5.4 Adaptive Online Learning

We design an experiment to evaluate the capability of adaptive online learning models in handling time series data under evolving data distributions. The experiment uniformly divides a time series dataset into 10 non-overlapping subsets to simulate incremental content. The first five subsets are used for the initial training of the model, followed by the gradual introduction of the remaining subsets for incremental learning. After the addition of each new subset, the model is evaluated. The experiment assesses the model's adaptability to new data and overall predictive performance by calculating the average of evaluation metrics across all incremental learning steps. This experiment aims to reveal the performance of online learning models in dynamic time series environments, especially under conditions where data distributions change over time. In this section, we select ELC and Weather datasets, and the baselines include Informer [13], Autoformer [14], MICN [57], and TimesNet [60]. For each subset of the data, we still adopt an experimental setting of $I, O = 96$, and use the last 20% of each subset as the test set.

Fig. 2 illustrates the performance of online learning on both ELC and Weather datasets. First, it can be observed that LAMEE's performance demonstrates a stable upward trend (i.e., a decrease in MAE) as the rounds of augmentation increase, indicating that LAMEE has excellent online learning capabilities and can adeptly adapt to data distributions evolving over time. Meanwhile, the other four baselines also generally show a trend of decreasing error, suggesting that with the addition of data and

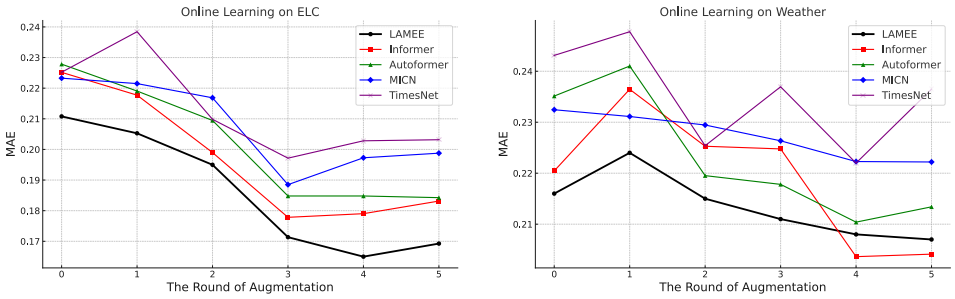

Fig. 2 The evaluation of adaptive online learning. The horizontal axis represents the number of rounds of incremental training conducted after each subset of data is added, where 0 indicates the baseline performance after training is completed on the first five datasets, and no new subsets have been added.

the integration of new data, all models can adapt well to time-evolving data distributions. However, the performance improvement of these baselines is not as stable and even carries the risk of declining in some rounds. This is because these complex models have a certain immunity to concept drift, which, although theoretically advantageous, can be a drawback for a system with strong time sensitivity. The sensitivity to inconsistencies in time series distribution, in fact, can be a beneficial trait. Thus, we see that LAMEE, with its simpler structure, can effectively perceive the concept drift brought about by data evolution over time, thereby better fitting the current data.

5.5 Interpretability

We analyze the weights of each layer to interpret the model, as we do not use high-dimensional embeddings for raw features. Using the Electricity dataset ($I = 96$, $O = 96$) as an example, we visualize the heatmaps of each layer in Figure 3.

5.5.1 Temporal Dense Layers

The first layer's scattered weights capture global info, with smaller weights for padded zeros and most info from input signals. The second layer's weights are widely distributed, emphasizing timestamps like the 4-th, 96-th, 124-th, and 148-th for their cyclic characteristics. In the third layer, the right half of the weight matrix, which corresponds to predictions, has larger weights than the left half, which is related to reconstructed input signals. This is because predicting future signals is harder than reconstructing input signals.

5.5.2 Frequency Dense Layers

For signals of length $96+96$ ($J+O$), we forward the lowest 32 frequency components. Since frequency dense layer weights are complex, we demodulate for visualization. Similar to temporal dense layers, the first frequency dense layer has a scattered weight distribution, with the lowest frequency components having the least weight. This is due

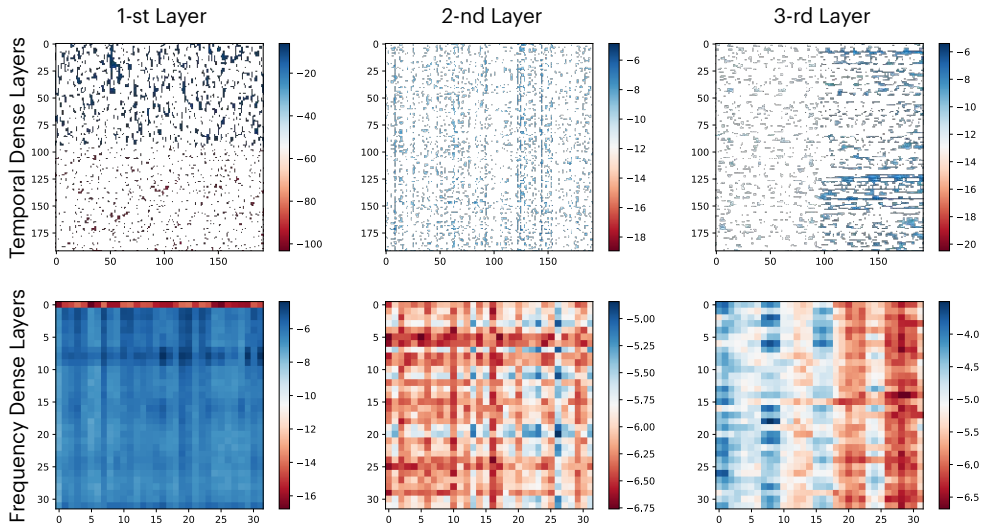

Fig. 3 Weights visualization for interpretability. We logarithmize all weights for better distinction, thus the visualization are relative values.

to the lowest frequency component in discrete Fourier transforms being 0, suggesting infinite seasonality. This component primarily influences trend, not the main focus of the frequency dense layers, thus having lower weights. The second layer assigns larger weights to the 4-th, 8-th, and 12-th frequency components, reflecting the inherent 48, 24, and 18-hour seasonality. The third layer prioritizes low-frequency components for determining overall prediction trends.

5.6 Parameter Sensitivity

LAMEE has three main settings: layer count (N), reconstructed input signal length (J) in input-auxiliary supervision, and passed frequency components (M). We tested these settings' impact on results using the Electricity and Weather datasets. In our tests, we used 96 timestamps to predict future 96, with $N \in \{2, 3, 4, 5\}$, $J \in \{24, 48, 72, 96\}$, and $M \in \{16, 32, 64, 96\}$.

5.6.1 Layers (N)

On the Electricity dataset, the best results come with 2-3 layers, adding more worse outcomes. But the Weather dataset does better with more layers. This is because the Electricity dataset has clear seasonal trends, which a few dense frequency layers can easily capture, and more layers could lead to overfitting. The Weather dataset, being non-stationary, benefits from more layers to better capture less obvious patterns.

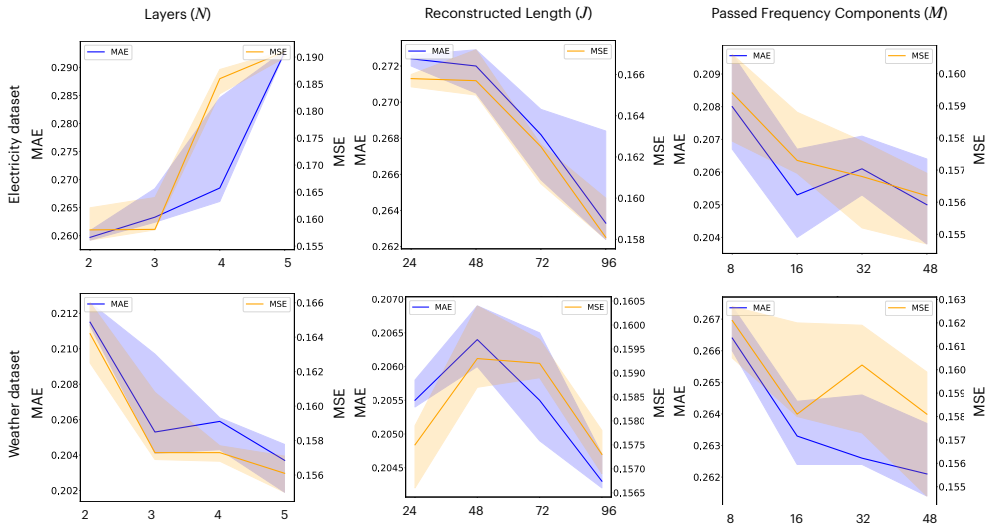

Fig. 4 Parameter sensitivity analysis.

5.6.2 The length of reconstructed input signals (J)

Our analysis of the length of reconstructed input signals shows a clear trend in both datasets: the best performance comes when hyper-parameters are at their highest values. The highest value for the length of reconstructed input signals means the entire input signal will be reconstructed, effectively capturing the characteristics provided by the input signals.

5.6.3 The passed frequency components (M)

The sensitivity analysis for the passed frequency components shows that best performance comes at the highest value, meaning no frequency components should be filtered out. However, this is less noticeable in the non-stationary Weather dataset compared to the stationary Electricity dataset. While increasing passed frequency components linearly increases computational complexity but only slightly improves performance. So, the model is most cost-effective when the passed frequency components are at 16% of the total length ($M = 16$).

5.7 Ablation Studies

This section evaluates different modules in our model: joint time-frequency information, residual decomposition, and input-auxiliary supervision. We study these through ablation models, namely LAMEE (w/o TDL), LAMEE (w/o FDL), LAMEE (w/o RD), and LAMEE (w/o IAS). The first two models test the joint time-frequency aspect by removing temporal or frequency dense layers. LAMEE (w/o RD) and LAMEE (w/o IAS) assess the effectiveness of residual decomposition and input-auxiliary supervision.

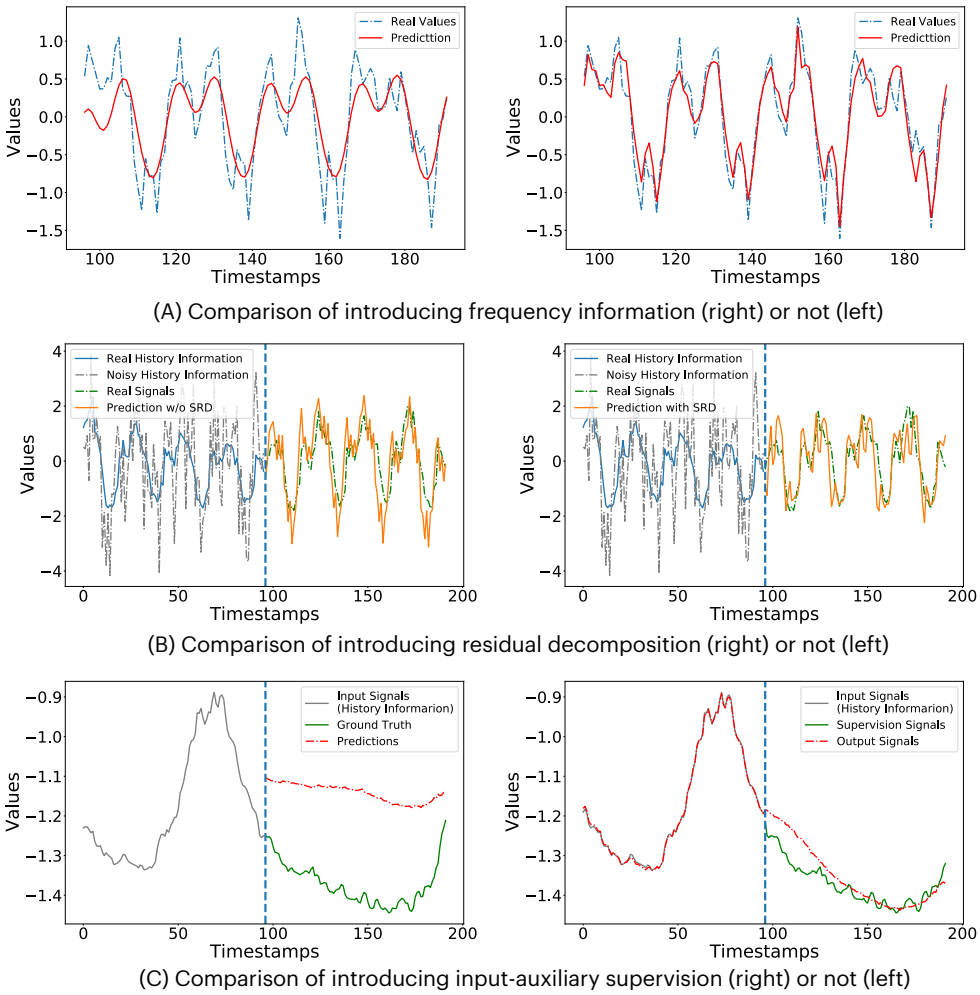

Fig. 5 Case visualization of ablation studies.

The results, displayed in Table 5 and Figure 5, suggest that each module contributes to performance enhancement. Furthermore, an incremental model, LAMEE (embeddings), shows that using raw features directly rather than high-dimensional embeddings can improve prediction accuracy.

5.7.1 Joint Time-Frequency Information

By introducing the frequency information, the prediction accuracy on all datasets will be increased by an average of 29.10%, especially on the Electricity dataset with strong seasonality, the performance will benefit even more (about 51.20%). This result confirms the correctness of our introduction of frequency information. By introducing the

Table 5 The results of ablation studies.

Models	Metrics	ILI		Electricity		Weather	
		24	36	92	196	92	196
LAMEE (w/o FDL)	MAE	0.709	0.841	0.476	0.483	0.240	0.280
	MSE	1.006	1.356	0.432	0.438	0.185	0.234
LAMEE (w/o TDL)	MAE	0.689	0.775	0.297	0.303	0.225	0.268
	MSE	0.943	1.312	0.185	0.186	0.176	0.229
LAMEE (w/o RD)	MAE	0.601	0.743	0.270	0.277	0.215	0.267
	MSE	0.849	1.320	0.168	0.190	0.160	0.233
LAMEE (w/o IAS)	MAE	0.710	0.779	0.280	0.322	0.218	0.281
	MSE	0.961	1.183	0.167	0.195	0.166	0.229
LAMEE (embeddings)	MAE	0.656	0.778	0.298	0.308	0.246	0.277
	MSE	0.893	1.359	0.179	0.213	0.173	0.245
LAMEE	MAE	**0.588**	**0.643**	**0.264**	**0.277**	**0.199**	**0.248**
	MSE	**0.832**	**1.083**	**0.158**	**0.176**	**0.153**	**0.203**

frequency information, the performance will also be improved, but the overall improvements in performance (12.20%) is smaller than introducing the frequency information, which might reflect that frequency information is more important than temporal information under such a case. In any case, this set of ablation studies demonstrates the importance and effectiveness of using joint time-frequency information. The corresponding case visualization is shown in Figure 5(A). Comparing Figure 5(A)(left) (without frequency information), high-frequency components can be better predicted by introducing frequency information, peaks and troughs can also be well fitted (Figure 5(A)(right)).

5.7.2 Residual Decomposition

While LAMEE (w/o RD) shows that residual decomposition only slightly improves metrics (7.0%), it's crucial for reducing error variance under extreme noise conditions, enhancing the model's prediction reliability. This is illustrated when we add Gaussian noise to the input signals. As seen in Figure 5(B), the complete LAMEE model (right) produces smoother and more accurate predictions with smaller error variance compared to LAMEE (w/o RD) (left), even with noisy signals, proving LAMEE's strong noise immunity and stability.

5.7.3 Input-Auxiliary Supervision

LAMEE (w/o IAS) shows the benefits of input-auxiliary supervision, it improves performance by about 11.0%. The effect of this strategy is visually shown in Figure 5(C), where the full LAMEE model's predictions align with the input signals' focus and trend, preventing trend inconsistent. This is especially effective for non-stationary cases, ensuring reliable predictions.

5.7.4 Embeddings

We also verified through LAMEE (embeddings) that directly using raw features for modeling and comparing features in high-dimensional embeddings resulted in a 14% performance improvement. This improvement primarily stems from the challenge

faced by MLPs in handling the non full-rank feature introduced by high-dimensional embeddings.

6 Conclusions

In this study, we introduced LAMEE, a novel lightweight All-MLP framework for Time Series Prediction Empowering Recommendations (TSPER). LAMEE has addressed several challenges and difficulties in the TSPER task concerning computational efficiency, model evolution, and performance assurance, resulting in a balanced time series prediction model in terms of performance, efficiency, and other aspects. Through extensive experiments, LAMEE demonstrated superior performance in various datasets, outperforming existing methods in both efficiency and effectiveness. Our work not only showcases the potential of LAMEE in enhancing time series prediction accuracy, but also opens new avenues for future research in this domain.

References

[1] Jannach, D., Zanker, M., Felfernig, A., Friedrich, G.: Recommender systems: an introduction (2010)

[2] Guo, Q., Zhuang, F., Qin, C., Zhu, H., Xie, X., Xiong, H., He, Q.: A survey on knowledge graph-based recommender systems. IEEE Transactions on Knowledge and Data Engineering **34**(8), 3549–3568 (2020)

[3] Sun, R., Cao, X., Zhao, Y., Wan, J., Zhou, K., Zhang, F., Wang, Z., Zheng, K.: Multi-modal knowledge graphs for recommender systems. In: Proceedings of the 29th ACM International Conference on Information & Knowledge Management, pp. 1405–1414 (2020)

[4] Huang, C., Xu, H., Xu, Y., Dai, P., Xia, L., Lu, M., Bo, L., Xing, H., Lai, X., Ye, Y.: Knowledge-aware coupled graph neural network for social recommendation. In: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 35, pp. 4115–4122 (2021)

[5] Sharaf, M., Hemdan, E.E.-D., El-Sayed, A., El-Bahnasawy, N.A.: A survey on recommendation systems for financial services. Multimedia Tools and Applications **81**(12), 16761–16781 (2022)

[6] Xue, J., Zhu, E., Liu, Q., Yin, J.: Group recommendation based on financial social network for robo-advisor. IEEE Access **6**, 54527–54535 (2018)

[7] Nilashi, M., Asadi, S., Minaei-Bidgoli, B., Abumalloh, R.A., Samad, S., Ghabban, F., Ahani, A.: Recommendation agents and information sharing through social media for coronavirus outbreak. Telematics and Informatics **61**, 101597 (2021)

[8] Hussain, M.M.-u., Avci, B., Trajcevski, G., Scheuermann, P.: Incorporating weather updates for public transportation users of recommendation systems. In:

2016 17th IEEE International Conference on Mobile Data Management (MDM), vol. 1, pp. 333–336 (2016). IEEE

[9] Djavadian, S., Hoogendoorn, R.G., Van Arerm, B., Chow, J.Y.: Empirical evaluation of drivers' route choice behavioral responses to social navigation. Transportation research record **2423**(1), 52–60 (2014)

[10] Hussain, M.M.-u., Avci, B., Trajcevski, G., Scheuermann, P.: Incorporating weather updates for public transportation users of recommendation systems. In: 2016 17th IEEE International Conference on Mobile Data Management (MDM), vol. 1, pp. 333–336 (2016). IEEE

[11] Yin, H., Zhou, X., Cui, B., Wang, H., Zheng, K., Nguyen, Q.V.H.: Adapting to user interest drift for poi recommendation. IEEE Transactions on Knowledge and Data Engineering **28**(10), 2566–2581 (2016)

[12] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, Ł., Polosukhin, I.: Attention is all you need. Advances in neural information processing systems **30** (2017)

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

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

[15] Zhou, T., Ma, Z., Wen, Q., Wang, X., Sun, L., Jin, R.: Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. International Conference on Learning Representations (2022)

[16] Liu, S., Yu, H., Liao, C., Li, J., Lin, W., Liu, A.X., Dustdar, S.: Pyraformer: Low-complexity pyramidal attention for long-range time series modeling and forecasting. In: International Conference on Learning Representations (2021)

[17] Lu, J., Liu, A., Dong, F., Gu, F., Gama, J., Zhang, G.: Learning under concept drift: A review. IEEE transactions on knowledge and data engineering **31**(12), 2346–2363 (2018)

[18] Zenisek, J., Holzinger, F., Affenzeller, M.: Machine learning based concept drift detection for predictive maintenance. Computers & Industrial Engineering **137**, 106031 (2019)

[19] Ding, C., Zhao, J., Sun, S.: Concept drift adaptation for time series anomaly detection via transformer. Neural Processing Letters **55**(3), 2081–2101 (2023)

[20] Zhao, Z., Xu, J., Zang, Y., Hu, R.: Adaptive abnormal oil temperature diagnosis method of transformer based on concept drift. Applied Sciences **11**(14), 6322 (2021)

[21] Trybulec, W.A.: Vectors in real linear space. Formalized Mathematics **1**(2), 291–296 (1990)

[22] Brand, L.: Vector and Tensor Analysis. Courier Dover Publications, ??? (2020)

[23] Arjovsky, M., Bottou, L.: Towards principled methods for training generative adversarial networks. arXiv preprint arXiv:1701.04862 (2017)

[24] Pallás-Areny, R., Webster, J.G.: Analog Signal Processing. John Wiley & Sons, ??? (1999)

[25] Orfanidis, S.J.: Introduction to Signal Processing. Prentice-Hall, Inc., ??? (1995)

[26] Box, G.E., Jenkins, G.M.: Some recent advances in forecasting and control. Journal of the Royal Statistical Society. Series C (Applied Statistics) **17**(2), 91–109 (1968)

[27] Box, G.E., Jenkins, G.M., Reinsel, G.C., Ljung, G.M.: Time Series Analysis: Forecasting and Control. John Wiley & Sons, ??? (2015)

[28] Gardner Jr, E.S.: Exponential smoothing: The state of the art. Journal of forecasting **4**(1), 1–28 (1985)

[29] Jalles, J.T.: Structural time series models and the kalman filter: a concise review (2009)

[30] Friedman, J.H.: Greedy function approximation: a gradient boosting machine. Annals of statistics, 1189–1232 (2001)

[31] Elsayed, S., Thyssens, D., Rashed, A., Jomaa, H.S., Schmidt-Thieme, L.: Do we really need deep learning models for time series forecasting? arXiv preprint arXiv:2101.02118 (2021)

[32] Wen, R., Torkkola, K., Narayanaswamy, B., Madeka, D.: A multi-horizon quantile recurrent forecaster. arXiv preprint arXiv:1711.11053 (2017)

[33] Rangapuram, S.S., Seeger, M.W., Gasthaus, J., Stella, L., Wang, Y., Januschowski, T.: Deep state space models for time series forecasting. Advances in neural information processing systems **31** (2018)

[34] Maddix, D.C., Wang, Y., Smola, A.: Deep factors with gaussian processes for forecasting. arXiv preprint arXiv:1812.00098 (2018)

[35] Oord, A.v.d., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., Kalchbrenner, N., Senior, A., Kavukcuoglu, K.: Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499 (2016)

[36] Borovykh, A., Bohte, S., Oosterlee, C.W.: Conditional time series forecasting with convolutional neural networks. arXiv preprint arXiv:1703.04691 (2017)

[37] Bai, S., Kolter, J.Z., Koltun, V.: An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. ArXiv **abs/1803.01271** (2018)

[38] Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)

[39] Ruck, D.W., Rogers, S.K., Kabrisky, M.: Feature selection using a multilayer perceptron. Journal of neural network computing **2**(2), 40–48 (1990)

[40] Hornik, K., Stinchcombe, M., White, H.: Multilayer feedforward networks are universal approximators. Neural networks **2**(5), 359–366 (1989)

[41] Krizhevsky, A., Sutskever, I., Hinton, G.E.: Imagenet classification with deep convolutional neural networks. Communications of the ACM **60**(6), 84–90 (2012)

[42] Hochreiter, S., Schmidhuber, J.: Long short-term memory. Neural Computation **9**, 1735–1780 (1997)

[43] Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: A survey. ACM computing surveys (CSUR) **54**(10s), 1–41 (2022)

[44] Tay, Y., Dehghani, M., Bahri, D., Metzler, D.: Efficient transformers: A survey. ACM Computing Surveys (CSUR) (2020)

[45] Ying, C., Cai, T., Luo, S., Zheng, S., Ke, G., He, D., Shen, Y., Liu, T.-Y.: Do transformers really perform bad for graph representation? In: Neural Information Processing Systems (2021)

[46] Wen, Q., Zhou, T., Zhang, C., Chen, W., Ma, Z., Yan, J., Sun, L.: Transformers in time series: A survey. arXiv preprint arXiv:2202.07125 (2022)

[47] Liu, R., Li, Y., Tao, L., Liang, D., Zheng, H.-T.: Are we ready for a new paradigm shift? a survey on visual deep mlp. Patterns **3**(7), 100520 (2022)

[48] Tolstikhin, I.O., Houlsby, N., Kolesnikov, A., Beyer, L., Zhai, X., Unterthiner, T., Yung, J., Steiner, A., Keysers, D., Uszkoreit, J., *et al.*: Mlp-mixer: An all-mlp architecture for vision. Advances in Neural Information Processing Systems **34**, 24261–24272 (2021)

[49] Melas-Kyriazi, L.: Do you even need attention? a stack of feed-forward layers does

surprisingly well on imagenet. arXiv preprint arXiv:2105.02723 (2021)

[50] Hendrycks, D., Gimpel, K.: Gaussian error linear units (gelus). arXiv preprint arXiv:1606.08415 (2016)

[51] Nussbaum, M.: Advanced digital signal processing and noise reduction. (2016)

[52] Persons, W.M.: Indices of business conditions: an index of general business conditions (1919)

[53] Zeng, A., Chen, M., Zhang, L., Xu, Q.: Are transformers effective for time series forecasting? arXiv preprint arXiv:2205.13504 (2022)

[54] Wu, Z., Pan, S., Long, G., Jiang, J., Chang, X., Zhang, C.: Connecting the dots: Multivariate time series forecasting with graph neural networks. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, pp. 753–763 (2020)

[55] Liu, Y., Wu, H., Wang, J., Long, M.: Non-stationary transformers: Exploring the stationarity in time series forecasting. In: Advances in Neural Information Processing Systems (2022)

[56] Woo, G., Liu, C., Sahoo, D., Kumar, A., Hoi, S.: Etsformer: Exponential smoothing transformers for time-series forecasting. arXiv preprint arXiv:2202.01381 (2022)

[57] Wang, H., Peng, J., Huang, F., Wang, J., Chen, J., Xiao, Y.: Micn: Multi-scale local and global context modeling for long-term series forecasting. In: The Eleventh International Conference on Learning Representations (2023)

[58] Zhang, T., Zhang, Y., Cao, W., Bian, J., Yi, X., Zheng, S., Li, J.: Less is more: Fast multivariate time series forecasting with light sampling-oriented mlp structures. arXiv preprint arXiv:2207.01186 (2022)

[59] Zhang, Y., Yan, J.: Crossformer: Transformer utilizing cross-dimension dependency for multivariate time series forecasting. In: The Eleventh International Conference on Learning Representations (2023)

[60] Wu, H., Hu, T., Liu, Y., Zhou, H., Wang, J., Long, M.: Timesnet: Temporal 2d-variation modeling for general time series analysis (2023)

Declarations

Ethical Approval

This study did not involve any experiments on human or animal subjects. Therefore, ethical approval was not required.

Funding

This work is funded in part by the National Natural Science Foundation of China Projects No. U1936213, and the Shanghai Science and Technology Development Fund No. 19DZ1200802.

Availability of Data and Materials

The data used in this study are publicly available. Details and access information for these data have been provided in the main text of the paper.