Applied Intelligence
https://doi.org/10.1007/s10489-023-05057-7

Multi-scale adaptive attention-based time-variant neural networks for multi-step time series forecasting

Gao Changxia[1] · Zhang Ning[1] · Li Youru[1] · Lin Yan[1] · Wan Huaiyu[1]

Accepted: 26 September 2023

Abstract
Time series analysis is the process of exploring and analyzing past trends to predict future events for any given time interval. Powered by recent advances in convolutional, recurrent and self-attention mechanisms, many deep learning methods have facilitated the investigation of time series forecasting. However, despite their effectiveness, it is doubtful that future trends can be accurately predicted due to the intricate temporal irregularities. Plus, time series frequently exhibit features at various time scales, but existing approaches do not adequately take this into account. To address above issues, this paper offers a new Multi-scale Adaptive attention-based Time-Variant neural Networks (MATVN) for multi-step ahead time series forecasting. Specifically, we contribute a novel framework capable of capturing irregular dynamic behaviors observed in temporal data over time with a Time-Variant architecture. Furthermore, a newly proposed Multi-scale Multi-head Adaptive attention module is introduced into the Time-Variant architecture to encode temporal dependencies from various pre-defined scale-aware ranges. Additionally, we endow the proposed module with more flexible individual representation learning and scale-aware attention scopes for each token to better capture multi-scale temporal patterns by designing a new Adaptive Window-aware Mask strategy. Experimental results on the vast majority of application scenarios, including climatology and energy consumption, demonstrate that the proposed model outperforms a lot of recent state-of-the-art methods in multi-step time series forecasting tasks.

Keywords Time series forecasting · Dynamic modeling · Multi-scale dependencies · Adaptive modeling

1 Introduction

Time series forecasting is an imperative ingredient across multiple critical areas, including traffic forecasting [1], disease propagation analysis [2] and weather forecasting [3]. Multi-step prediction problem describes the complete trajectory of predicting future sequence values over a relatively long-time horizon instead of reducing future prediction to a single value, which differs from one-step prediction problem [4]. Therefore, multi-step ahead time series forecasting requires an accurate description of time series evolution.

However, over the past decade, existing methods such as vector auto-regression (VAR) [5] and seasonal autoregressive sliding average model (SARIMA) [6] have been designed to solve time series forecasting problems by effectively transforming non-stationary process into stationary one through differencing. However, due to the stationary assumption, they become ineffective for other numerous real-world time series data with abrupt changes of distribution.

Recently, there has been a greater focus placed on deep learning methods within the time series forecasting problem [7]. Of particular concern in the area are Recurrent Neural Networks-based (RNNs-based) approaches, which are beneficial to capture time-series patterns due to their recurrent structures that make the current hidden state highly relevant to the previous one and the current input [8]. Convolutional Neural Networks (CNNs)[9] become another popular alternative that extract complex features by utilizing convolutional filters in parallel. As a third category of effective and commonly used tool, self-attention has inspired and broadened new ideas for researchers to model time series. Self-attention assists Transformer [10] in accessing any part of history data without being affected by distance, making it more effective in discovering long-term recurring patterns. Taken together, approaches mentioned above do

✉ Wan Huaiyu
hywan@bjtu.edu.cn

[1] School of Computer and Information Technology, Beijing Jiaotong University, 100044 Beijing, China

Published online: 20 October 2023

achieve promising forecasting results, however, they suffer from inadequate framework design in terms of the following two aspects:

Time-invariant architecture Over a set of time steps, multiple components such as noise, seasonality and irregular complex temporal patterns may be observed within a certain range of scales [11]. It is noteworthy that the irregular scale range that needs to be modeled will increase as the number of forecast steps increases. Therefore, we speculate that if the parameter and architecture of a model can be adjusted accordingly with the irregular changes of these scales, it will generate more accurate prediction results. However, RNNs fail to achieve this goal because its parameter sharing mechanism impedes the model's capacity to learn time-varying relationships [12]. Specifically, the most commonly used metaphor in describing RNNs is that of (nonlinear) dynamical system. The dynamics, in general, describe how a physical quantity (time series observation value X_t) changes over time. For time series forecasting, the most important task is how to model this nonlinear mapping function or the dynamics f. To model the dynamics, RNNs maintain a vector h_t that summarizes past observations and is updated as $h_t = tanh(UX_t + Wh_{t-1})$. From the formula, we can see that the dynamics of RNNs are still time-invariance since parameters U and W are not determined by inputs but are fixed and replicated over time [13], which is also shown in Fig. 1(a). Similar cases also occur in CNNs. Be aware that each filter of CNNs is replicated across the overall field. The replicated units share the same weight vector and generate a feature map, meaning all neurons in given convolutional layer can be responsive to the same feature in their particular visual response field [14]. This parameter sharing mechanism in CNNs leads to a shift-invariance model in spatial domain, which translates to time-invariance in time series applications [15]. We conjecture that the time-invariance makes the model less effective to perform multi-step ahead time series forecasting task than one whose parameters change over time shown in Fig. 1(b) due to the lack of capacity to capture complex irregular temporal patterns. This issue is extensively addressed in [16]. However, prediction performance still needs to be improved because it does not sufficiently take into account the characteristics of time series on various time scales.

Fixed-size scopes Multi-scale information is important for time series modeling [17]. It is well known that many phenomena and processes in the real world often have patterns of change over multiple time scales. This multi-scale information is reflected in time series, which can be analyzed and observed at different time scales to better understand and explain the behavior and characteristics of the time series. For example, meteorological phenomena are often subject to by climate change at different scales such as daily, seasonal, and annual. In this case, we speculate that if a model can offer greater flexibility in adjusting the attention scopes based on the sequence dynamics, it can effectively capture the variations in the sequence at different scales, allowing for more accurate identification of daily, seasonal, and yearly patterns of change within the data. However, the structure of CNNs

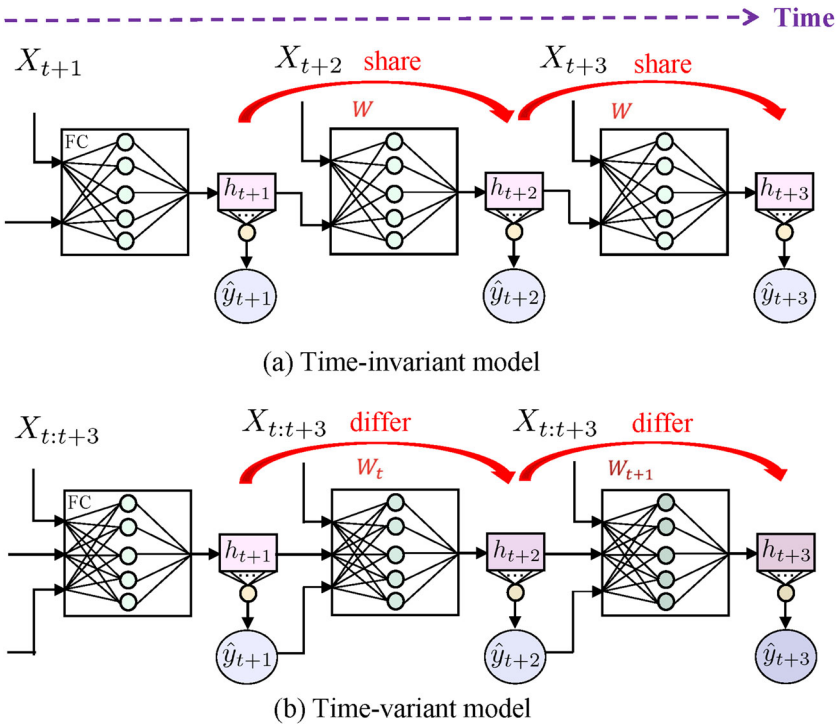

Fig. 1 An illustration of a simple example of Time-Variant architecture in which hidden cells are equipped with fully connected layer. Of note, although hidden cells (modules) in Time-Variant architecture are the same, the weights W_t differ in the timeline direction. This is different from RNN, whose cell shares the same weight W with the next cell along the time

does not meet such framework requirements because it convolves fixed-size filters along input sequence to provide some clues about time series trends currently [15]. Employing such convolutional filters makes it difficult for CNNs to alter to arbitrary scales by conditioning their receptive fields based on their input sequences, resulting in a lack of flexibility in adapting to local time series dependencies. Some attention-based improvements [18, 19] are proposed to address this issue. They incorporate attention mechanisms into CNNs to select representative feature information extracted from different convolutional layers to dynamically construct task-friendly local relations. Filters have fixed sizes, so multiple convolutional layers are required to stack to get different sizes of receptive fields. Therefore, it is inefficient due to the unavoidable enumeration of all possible kernel sizes. Similarly, RNNs also do not recognize data's local characteristics at variable scales because RNNs use a fixed-size hidden state to store the sequence information [20]. Moreover, as a widely investigated alternative way, the self-attention can capture global dependencies of sequences without consideration of distances by taking into account all positions in input signals [10]. And its local-oriented variants [21–24] restrict the attention scope from the global area to a local one to capture local patterns or multiple scale patterns in time series. However, they are limited by employing pre-defined fixed-size scopes, which raises the same issue as CNNs using fixed filters. Therefore, it appears from the evidence provided above that these studies cannot flexibly learn adaptive sizes of local features for sequential modeling.

In this paper, we attack the above issues using the following strategies. A novel algorithm termed as Multi-scale Adaptive attention-based Time-Variant neural Networks (MATVN) is proposed in this paper for multi-step time series forecasting. The innovative idea of the proposal lies in its Time-Variant architecture, which is committed to characterizing the time series data in a dynamic manner and obtaining more accurate forecasting results compared to the typical CNNs-based and RNNs-based frameworks. Building on the time-variant capability of MATVN, we design a Multi-scale Multi-head Adaptive attention module to encode temporal dependencies from different scale ranges. In addition, the ability to obtain sequential information is further improved by flexibly configuring different sizes of attention ranges for each token in time series.

In general, we make the following main contributions:

1. We aim at formulating our multi-step ahead time series architecture in a novel time-variant scheme to learn dynamic irregular temporal behavior over time.
2. We present a Multi-scale Multi-head Adaptive attention module capable of capturing scale-aware dependencies. The proposed module encodes time series information on different temporal scales.
3. In addition, a new Adaptive Window-aware Mask strategy is also provided to make the proposed module capture diverse dependencies for time-series forecasting tasks. Instead of choosing a suitable fixed attention range representation for each token (the value corresponding to each time point) in sequential data, the proposed module can dynamically select the attention scopes so as to focus only on the part that is relevant to the current input, which improves the flexibility and generalization ability of the model.
4. A series of experiments conducted on three benchmark time series datasets show remarkable improvements with the proposed approach compared to a large set of baselines.

The rest of the paper is organized as follows. Section 2 reviews some related work. Section 3 presents the problem definition and symbol notation and then describes the proposed approach. Section 4 analyzes the experimental results. Finally, Section 5 provides the conclusion and future work.

2 Related work

A variety of models have been developed to improve time series forecasting accuracy. For example, Hajirahimi et al. [25] proposes a hybrid model and concludes that combining hybrid models leads to superior performance compared to using a standalone model. Nevertheless, it is essential to consider that the hybrid approaches typically require the computation and integration of multiple prediction models, which increases the complexity and computational cost of the method. For some scenarios where there are ordered patterns in datasets, Wu et al. [26] delves into the order-preserving rule (OPR) mining and introduces the order-preserving rule miner (OPR-Miner) algorithm, designed to uncover robust rules from all frequent order-preserving patterns (OPPs), so as to improve the prediction results of regular sequences. More recently, the dominant direction in modeling time series forecasting tasks has been deep learning approaches. Comparatively speaking, deep learning methods utilize neural networks and other models to make predictions by learning features directly from the data without relying on pre-defined patterns. As a result, deep learning has been widely used in areas such as probabilistic prediction and point prediction, and excels in handling complex patterns of large-scale datasets. For example, Chen et al. [27] introduces a versatile and adaptable deep hybrid graph-based probabilistic forecasting framework. This framework employs a relational global model to capture complex non-linear patterns globally by considering the dependencies between time-series in the graph. Simultaneously, it utilizes a relational local model to capture the individual random effects of each time-series

locally. For point prediction, common deep learning methods can be broadly categorized into the following three groups:

Recurrent neural networks RNNs can significantly improve prediction performance by storing historical information with recurrent units. Ilhan et al. [28] introduces a new nonlinear regression RNN structure that can adaptively switch between internal regions in a Markov way and be used for non-stationary modeling of sequence data. Cirstea et al. [29] designs an architecture for enhancing correlated time series forecasting, in which RNN aims to capture the dynamics of time series. Although these studies have yielded remarkable forecasting results on numerous sequential tasks, theoretical evidence suggests that RNNs raise vanishing/exploding gradient problems very easily when predicting a very long sequence. In addition, once RNNs unfold with the time dimension, we regard it as a time-invariant neural network due to its parameter sharing mechanism, which hinders its ability to describe the dynamics over time in complex sequences. Furthermore, RNNs employ hidden states with fixed-size to model time series dependencies, making it difficult to learn variable sizes of sequential information for different tokens efficiently.

Convolutional neural networks There is a growing body of literature that recognizes the importance of CNNs for sequence learning. In comparison with the chain structure in RNNs, CNNs capture temporal dependencies by stacking multiple layers without any recurrent functions. Bai et al. [30] compares and evaluates the convolutional and recurrent frameworks, experimental results demonstrate that a simple convolution framework exhibits better performance than a recurrent framework on different tasks. Liu et al. [31] reports a new design based nonpooling CNN to extract both real and simulated sequential data with trends and seasonality. Tang et al. [32] proposes a novel CNN-based approach named Omni-Scale CNNs to cover appropriate receptive field size across different datasets for time series classification. Despite CNNs' success in parallel computing, it's important to be aware of their following limitations. Similar time-invariant cases also occur in CNNs, which restricts the capacity of the model to encode information in a dynamic manner. It may be problematic for CNNs to flexibly learn adaptive sizes of local features because it convolves along inputs with fixed filters.

Self-attention mechanism More recently, the existing literature on self-attention is extensive and focuses particularly on machine translation [33], sequential recommendation [34] and time series forecasting [35]. Wu et al. [36] designs a novel decomposition architecture named Autoformer, in which Auto-Correlation mechanism aims to capture time dependencies at subsequence level. Wu et al. [37] utilizes Transformer to capture long-term patterns from temporal data to forecast influenza-like illness (ILI). Zhou et al. [38] introduces a novel frequency-enhanced Transformer to improve its capability to perform time series forecasting by combining the seasonal-trend decomposition method with Transformer. These studies have demonstrated that self-attention performs a supervisory role in sequence modeling. A major limitation with this type of method, however, is that canonical dot-product self-attention is insensitive to local patterns. This main reason can be attributed to the fact that it is difficult for self-attention to disperse the distribution of attention because all the signals are considered simultaneously by weighted averaging. Such structural flaw in self-attention will bring underlying optimization issues to time series forecasting because in time series forecasting task, the data close to each other will have a great impact on each other [39]. Li et al. [23] aims to mitigate this problem by limiting the attention region to a local window using causal convolution. However, it is also noteworthy that the local-oriented method explores local temporal patterns with a fixed scope, making flexible modeling and accurate prediction of time series impossible.

We suggest the following three ways to mitigate the aforementioned issues. 1): We aim at formulating our multi-step ahead time series architecture in a novel time-variant scheme to model dynamics over time. 2): To empower time-variant architecture to encode time information from multiple-scale perspectives, we explore a novel module based on self-attention. 3): In addition, a new strategy is developed to give the proposed module more flexibility in selecting the most important scale for each token in time series.

3 Methodology

In this section, the problem definition and symbol notation are given first. Then we introduce the overview of the proposed model. Next, we elaborate on Time-Variant architecture and its important components, including Input, Multi-scale Multi-head Adaptive attention module, and Output. Furthermore, we also provide the proof of time-variance of proposed model. At last, we provide the loss function that our model aims to minimize.

3.1 Preliminary

Multi-Step Time Series Forecasting. Given a series of historical observations $X_{1:t} = \{X_1, X_2, \ldots, X_t \mid X_t \in \mathbb{R}^{d_x}\}$ recorded sequentially in time, where $X_t \in \mathbb{R}^{d_x}$ is a value at time t and d_x is the last dimension of input data. Our goal is to forecast multi-step time series values $\hat{Y}_{t+1:t+\tau} = \{\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+\tau} \mid \hat{y}_t \in \mathbb{R}^{d_y}\}$ between time $t+1$ and

time $t+\tau$, where $\hat{y}_t \in \mathbb{R}^{d_y}$ is the prediction at time t, τ is defined as the number of prediction steps and d_y is the ouput dimension. We define the problem formulation as follows:

$$\hat{Y}_{t+1:t+\tau} = f(X_{1:t}; \Theta) \tag{1}$$

where $f(\cdot)$ represents a nonlinear mapping function and Θ denotes the learnable parameter.

Output dimension. Note that the dimension of the output variable d_y is not limited to univariate. Depending on whether d_y is greater than 1, we categorize the forecasting problem into univariate prediction and multivariate prediction.

3.2 Overall framework

We design a novel framework called Multi-scale Adaptive attention-based Time-Variant neural Networks (MATVN) in this paper. Figure 2 outlines its overall framework. From it, we can see that different from [16], we primarily improve the performance of multi-step time series forecasting from the following two aspects: 1) Construct a novel Time-Variant architecture, MATVN, to flexibly capture irregular dynamic temporal behavior over time. 2) Design a new Multi-scale Multi-head Adaptive attention (MMA) module to learn temporal dependencies from different pre-defined scale-aware ranges, in which Adaptive Window-aware Mask strategy makes MMA extract features at various scales adaptively, resulting in superior multi-scale feature representation.

3.3 Time-variant architecture

We contribute a novel time-variant neural network capable of learning irregular dynamics observed in time series with its Time-Variant architecture (TV architecture). Figure 2 depicts its overall architecture. It is essentially a deep feed-forward architecture with interleaved outputs, which mainly consists of a series of inputs, a series of hidden modules, and a series of outputs. In a feed-forward network, although the architecture of each hidden module is identical, each module has its own unique parameter set. That is, the network between interleaved outputs varies in terms of parameter and architecture over time. Thus, the result is a time-variant model. In the following paragraphs, we will introduce the building parts of TV architecture MATVN and its time-variant proof in detail.

Input We transmit a set of univariate/multivariate sequence data $X_{1:t} = [X_1, X_2, ..., X_t] \in \mathbb{R}^{B \times L \times d_x}$ to each hidden module in the middle of TV architecture, where B denotes the batch size, L represents the length of sequence data input into the network, d_x is the dimension of input data, which is equal to 1 when input data is univariate.

Multi-scale multi-head adaptive attention As depicted in Fig. 3, a novel self-attention module is designed to adaptively capture dependencies of different attention ranges for each token from multiple scales. The proposed module mainly consists of four parts, including an input linear layer, a positional encoding scheme, N encoder layers, and an output

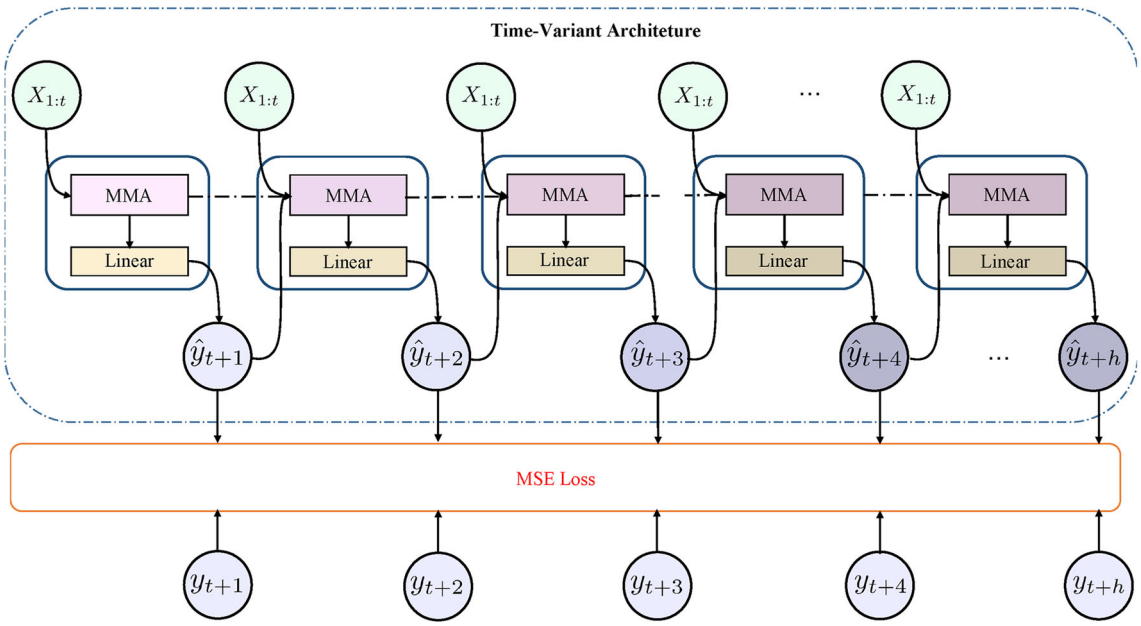

Fig. 2 Graphical illustration of the MATVN framework. The overall framework is a Time-Variant architecture, which is primarily embodied in network structure that differs in parameters over time. Here, a series of hidden modules sequentially connected in the middle of the architecture are illustrated with different colors to denote varying network framework and parameter over time

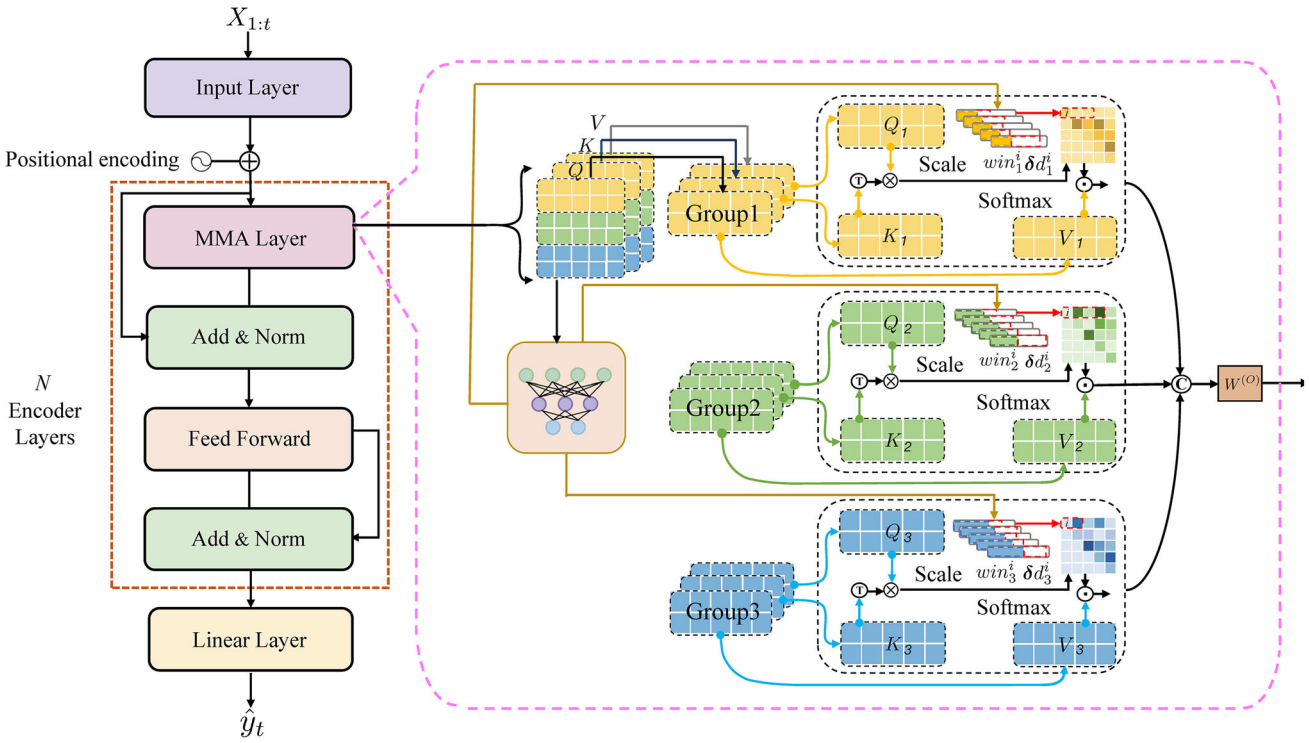

Fig. 3 An illustration of MMA module, what is striking in the figure is MMA layer in purple dotted box. It is clear from this figure that each attention matrix in each head is restricted by adaptive windows, which are formed by adding offsets learned from data to three pre-defined fixed windows, respectively. This successfully turns Multi-scale Multi-head Diverse attention (MMD) module into MMA, which enables more flexible multi-scale time series modeling

linear layer. It is noteworthy that there are two sub-layers, i.e., a proposed self-attention sub-layer and a feed-forward sub-layer in each encoder layer. Upon the output of the two sub-layers, residual connection and normalization are stacked.

The main goal of the input linear layer is to convert input tensor $X_{1:t} = [X_1, X_2, \cdots, X_t] \in \mathbb{R}^{B\times L\times d_x}$ into a higher dimension tensor $X_{1:t} = [X_1, X_2, \cdots, X_t] \in \mathbb{R}^{B\times L\times d_{in}}$ for enhancing data representation capacity. Owing to no recurrence and no convolution structure in the self-attention mechanism, a smart positional encoding scheme is also added to encode sequential information.

Then the converted tensor $X_{1:t} = [X_1, X_2, \cdots, X_t] \in \mathbb{R}^{B\times L\times d_{in}}$ is transmitted into N encoder layers. From Fig. 3, we can see that the proposed module and the canonical Multi-head Self-Attention (MSA) differ in the layout of the MMA layer, which is in charge of capturing local temporal patterns for each token in a time series from different scale-aware perspectives using the learnable window. In the following parts, we first introduce the canonical MSA mechanism and then elaborate on our proposed mechanism.

Broadly speaking, the essence of the MSA mechanism is to allow the attention function to model richer dependencies from different representation subspaces instead of a single one. It can be formalized as: Given a time series tensor $X_{1:t} = [X_1, X_2, \cdots, X_t] \in \mathbb{R}^{B\times L\times d_{in}}$, with length L and dimension d_{in}, then the multi-head self-attention projects $X_{1:t}$ into query Q_h , key K_h , and value V_h with $h = 1, 2, \cdots H$, where H is the total number of attention heads. After linear projection, the attention matrix Att_h and the output head_h of h-th head are defined as follows, respectively.

$$\text{Att}_h = \text{Attention}(Q_h, K_h) = \text{softmax}\left(\frac{Q_h K_h^T}{\sqrt{d}}\right), \tag{2}$$

$$\text{head}_h = (\text{Att}_h)\, V_h, \tag{3}$$

$$Q_h = W_h{}^{(Q)} X_{1:t},\ K_h = W_h{}^{(K)} X_{1:t},\ V_h = W_h{}^{(V)} X_{1:t}, \tag{4}$$

where d is the scaling factor, $W_h^{(Q)}$, $W_h^{(K)}$ and $W_h^{(V)}$ are learnable parameters. Afterwards, $\text{head}_1, \cdots,$ head_H are concatenated and projected by learnable parameter $W^{(O)}$ again as follows:

$$\text{MSA}(X_{1:t}) = [\text{head}_1, \cdots, \text{head}_H] W^{(O)}. \tag{5}$$

As indicated above, MSA is a mechanism for relating all the positions of the whole input sequence to get an attention weight matrix, which facilitates long-term dependencies learning. However, in this form, it is a failure to disperse the attention distribution, and then difficulty arises when an attempt is made to capture local dependencies. Technically, the simplest way to address this problem is to apply a window mask to each token in the self-attention weight matrix, limiting it to focus on a set of surrounding representations. We let the window size of each token i of attention weight matrix be win^i. This indicates that attention is only activated at the position where the element is in the window. The position with negative infinity outside the window is canceled.

Formally, the local window attention mask is defined as follows:

$$M(win^i)_{a,b} = \begin{cases} 0, & \text{if} |a-b| \leq win^i \quad \text{and} \quad b \geq a \\ -\infty, & \text{otherwise} \end{cases}, \tag{6}$$

where $a, b \in \{0, 1, \ldots, L-1\}$ are positional indexes, and $i \in \{1, 2, \ldots, L\}$ represents the i-th row of the matrix. Then, we add it to canonical attention matrix to get the output of Single-scale Multi-head Fixed attention (SMF) module:

$$\text{SMF}(X_{1:t}, win^i) = [\text{head}_1(X_{1:t}, win^i), \cdots, \text{head}_H(X_{1:t}, win^i)]W^{(O)}, \tag{7}$$

where

$$\begin{aligned} \text{head}_h(\cdot) &= (\text{SMF-Att}_h)V_h \\ &= \text{softmax}\left(\frac{Q_h {K_h}^T}{\sqrt{d}} + M\left(win^i\right)_{a,b}\right)V_h. \end{aligned} \tag{8}$$

We can deduce from (8) that when the element in the mask $M(win^i)_{a,b}$ is negative infinity, the value at the corresponding position in head_h is zero after calculating softmax, successfully overshadowing and suppressing the global attention weight of standard self-attention. Nevertheless, we believe that if a fixed-size window is relaxed to multiple fixed-size windows with different sizes located in different heads, self-attention can discover richer multi-scale sequential dependencies [21].

To realize the above idea, we investigate a Multi-scale Multi-head Diverse attention (MMD) module. Figure 3 illustrates the architecture diagram of our proposal. We design three groups of fixed-size windows $Win = [win^i_1, win^i_2, win^i_3]$ and let $H = 3$. Each element win^i_h indicates the attention scope on which various tokens i in the attention matrix of h -th head can focus. In this manner, we can extract information from multiple scales on different heads. Mathematically, we hence get the following attention mask and the output of MMD:

$$M(win^i_h)_{a,b} = \begin{cases} 0, & \text{if } |a-b| \leq win^i_h \quad \text{and} \quad b \geq a \\ -\infty, & \text{otherwise} \end{cases}, \tag{9}$$

$$\begin{aligned} &\text{MMD}(X_{1:t}, Win) \\ &= [\text{head}_1(X_{1:t}, win^i_1), \cdots, \text{head}_H(X_{1:t}, win^i_h)]W^{(O)}, \end{aligned} \tag{10}$$

where

$$\begin{aligned} \text{head}_h(\cdot) &= (\text{MMD-Att}_h)V_h \\ &= \text{softmax}\left(\frac{Q_h {K_h}^T}{\sqrt{d}} + M\left(win^i_h\right)_{a,b}\right)V_h. \end{aligned} \tag{11}$$

Furthermore, in order to empower MMD with the flexibility to capture deeper multi-scale sequence dependencies, an Adaptive Window-aware Mask is designed to allocate a learnable window rather than a fixed-size one for each token. As described in Fig. 3, we take fixed-size windows as reference, and then predict Adaptive Window-aware Offsets δD on the reference, depending on input feature. Of note, in δD, each element δd^i_h denotes the learned offset of a fixed window size for i-th token in sequence in h-th head. In this way, each token is given an adjustable, learnable window size rather than providing all tokens with fixed attention scopes. Mathematically, the Adaptive Window-aware Vector $AWin$ is formulated as follows:

$$\begin{aligned} AWin &= Win + \delta D \\ &= Win + \sigma \text{E}(Q) * L \\ &= Win + \sigma((W_1 Q + b_1)W_2 + b_2) * L, \end{aligned} \tag{12}$$

where W_1, W_2, b_1 and b_2 are trainable parameters. Q represents the query before grouping. $\text{E}(\cdot)$ can be defined as any feature extractor, and in our design, we utilize a single two-layer feed-forward sub-network. After that, a sigmoid activate function σ whose value ranges from zero to one is followed to predict offset. It is important to note that our module degenerates to MMD when the offset is equal to zero.

We generate an Adaptive Window-aware Mask $M(win^i_h + \delta d^i_h)_{a,b}$ after obtaining the above $AWin$ as follows:

$$M(win^i_h + \delta d^i_h)_{a,b} = \begin{cases} 0, & \text{if} |a-b| \leq win^i_h + \delta d^i_h \quad \text{and} \quad b \geq a \\ -\infty, & \text{otherwise} \end{cases}, \tag{13}$$

Then we add $M(win_h^i + \delta d_h^i)_{a,b}$ to (3) to provide a more flexible strategy for the self-attention mechanism. Formally,

$$\begin{aligned}\text{head}_h(\cdot) &= (\text{MMA-Att}_h)V_h \\ &= \text{softmax}\left(\frac{Q_h {K_h}^T}{\sqrt{d}} + M\left(win_h^i + \delta d_h^i\right)_{a,b}\right)V_h.\end{aligned} \tag{14}$$

Finally, the output of MMA can be denoted as follows:

$$\begin{aligned}&\text{MMA}(X_{1:t}, Win + \delta D) = [\text{head}_1(X_{1:t}, win_1^i + \delta d_1^i), \cdots, \\ &\text{head}_H(X_{1:t}, win_h^i + \delta d_h^i)]W^{(O)}\end{aligned} . \tag{15}$$

Upon the output of the MMA layer, we add the feed-forward layer to improve the expressiveness of the model. It is well known that the feed-forward layer mainly consists of two fully connected layers, between which is the ReLu activation function. As illustrated in Fig. 3, its input can be expressed as follows:

$$FFC_{in} = \text{LayerNorm}(\text{MMA}(X_{1:t}, Win + \delta D) + X_{1:t}), \tag{16}$$

where $FFC_{in} \in \mathbb{R}^{B\times L\times d_{in}}$ is the value after preforming Add&Norm on the output of MMA layer, $\text{MMA}(X_{1:t}, Win+\delta D) \in \mathbb{R}^{B\times L\times d_{in}}$ and input tensor $X_{1:t} \in \mathbb{R}^{B\times L\times d_{in}}$. LayerNorm denotes a form of normalization operation. And its output can be formulated as follows:

$$FFC_{out} = \text{ReLu}\left((FFC_{in})W_3 + b_3\right)W_4 + b_4, \tag{17}$$

$$\text{ReLu}(x) = \max(0, x), \tag{18}$$

where $FFC_{out} \in \mathbb{R}^{B\times L\times d_{in}}$ represents the transformed space of FFC_{in}, $W_3 \in \mathbb{R}^{d_{in}\times d_f}$, $b_3 \in \mathbb{R}^{d_f}$, $W_4 \in \mathbb{R}^{d_f\times d_{in}}$, $b_4 \in \mathbb{R}^{d_{in}}$ are learnable parameters. d_f is the transformed dimension through the first fully connected layer. In our design, we set $d_f = 3d_{in}$.

Then, an output linear layer is designed to produce the expected prediction as follows:

$$\hat{y}_t = (F\hat{F}C_{out})W_5 + b_5, \tag{19}$$

where $\hat{y}_t \in \mathbb{R}^{B\times d_y}$ denotes the predicted value at time t, $F\hat{F}C_{out}$ is a transformed value by performing Add&Norm on $FFC_{out} \in \mathbb{R}^{B\times L\times d_{in}}$. It is worth noting that the shape of $F\hat{FC}_{out}$ is $F\hat{FC}_{out} \in \mathbb{R}^{B\times d_{in}}$ because we first reshape $FFC_{out} \in \mathbb{R}^{B\times L\times d_{in}}$ into $FFC'_{out} \in \mathbb{R}^{B\times d_{in}\times L}$, then $FFC'_{out} \in \mathbb{R}^{B\times d_{in}\times L}$ is transformed into $F\hat{F}C_{out} \in \mathbb{R}^{B\times d_{in}}$ by the last linear layer of MMA module. Finally, we get the final prediction result $\hat{y}_t \in \mathbb{R}^{B\times d_y}$ after linear transformation again, $W_5 \in \mathbb{R}^{d_{in}\times d_y}$ and $b_5 \in \mathbb{R}^{d_y}$ are learnable parameters.

Output The forecast $\hat{Y}_{t+1:t+\tau} = \left[\hat{y}_{t+1}, \hat{y}_{t+2}, \ldots, \hat{y}_{t+\tau}\right]$ is provided by transmitting inputs (circles) to MATVN. Of note, each hidden module is designed to generate a forecast. We all know that the more output, the more likely the network is to arise the vanishing gradient problem. In our design, we alleviate this problem by having the structure interleave the outputs between hidden modules, which decomposes the chain into the sum of a series of factors instead of their product, thus leading to a more stable network and more accurate prediction results.

The time-variance of MATVN We detail the components of our framework in the above paragraphs. To better illustrate that our model is time-variance, a specific proof is provided in the following part.

Theorem 1 *As illustrated in Fig. 4, MATVN whose input is X_t, hidden state is h_t and forecast is $\hat{y}_t$ at time t as follows:*

$$h_t = MMA_t\left(X_t, h_{t-1}, \hat{y}_{t-1}\right), \tag{20}$$

$$\hat{y}_t = \text{L}_t(h_t), \tag{21}$$

is a time-variant model, where MMA_t is a MMA module and L_t is a linear layer at time t. h_{t-1} and $\hat{y}_{t-1}$ are hidden state and prediction at time $t-1$, respectively. We combine (20) and (21) to deduce prediction $\hat{y}_t$ at time t as follows:

$$\hat{y}_t = L_t(MMA_t\left(X_t, h_{t-1}, \hat{y}_{t-1}\right)). \tag{22}$$

Proof *Given input X_t and hidden state h_t, we shift them by t_0 respectively to get shifted value of X_t, $X'_t = X_{t-t_0}$ and shifted value of h_t, $h'_t = h_{t-t_0}$. In a time-invariance system, time shift of input sequence leads to corresponding shift in output sequence [40]. Thus, it requires that $\hat{y}_{t-t_0} = \hat{y}'_t$ if system is time-invariance. According to (22), $\hat{y}_{t-t_0}$ and $\hat{y}'_t$ can be expressed as follows:*

$$\hat{y}_{t-t_0} = L_{t-t_0}(MMA_{t-t_0}\left(X_{t-t_0}, h_{t-t_0-1}, \hat{y}_{t-t_0-1}\right)), \tag{23}$$

where $\hat{y}_{t-t_0}$ is the predicted value corresponding to input X_{t-t_0}, h_{t-t_0-1} is the hidden state and $\hat{y}_{t-t_0-1}$ is the predicted value of previous one time step, respectively.

$$\begin{aligned}\hat{y}'_t &= L'_t(MMA'_t(X'_t, h'_{t-1}, \hat{y}'_{t-1})) \\ &= L'_t(MMA'_t(X_{t-t_0}, h_{t-t_0-1}, \hat{y}'_{t-1})),\end{aligned} \tag{24}$$

where $\hat{y}'_t$ is the predicted value corresponding to input X'_t, h'_{t-1} is the hidden state and $\hat{y}'_{t-1}$ is the predicted value of previous one time step, respectively. The MMA module and

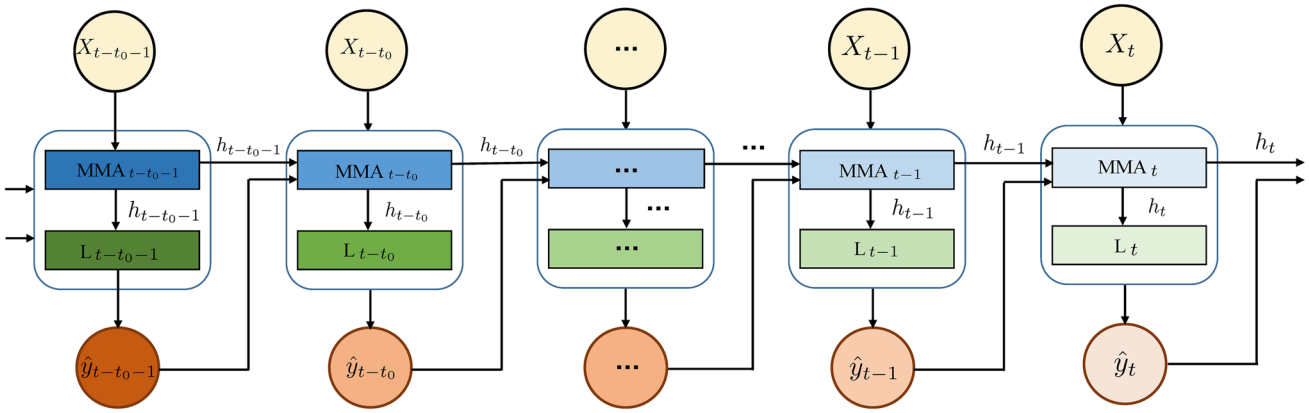

Fig. 4 An illustration of the time-variance of MATVN. The network between interleaved outputs varies in terms of parameter and architecture over time. Thus, MMA module and output layer are presented by different colors to indicate heterogeneity in the sequence

output layer also become MMA'_t *and* L'_t *according to input* X'_t*. Through above reasoning, we conclude that MATVN is time-variant because* $\hat{y}_{t-t_0} \neq \hat{y}_t{}'$*.* □

The input signal delay of MATVN not only shifts output sequence in time, but also changes network parameter and behavior. This differs from RNN in which the fixed parameters are reused at each time step. Therefore, we compare and analyze the time invariance of RNN, which is conducive to a better understanding of the time-variant structure of MATVN.

Theorem 2 *RNN whose input is* X_t*, hidden state is* h_t *and forecast is* $\hat{y}_t$ *at time* t *shown below:*

$$h_t = F(X_t, h_{t-1}), \tag{25}$$

$$\hat{y}_t = L(h_t), \tag{26}$$

is a time-invariant model, where F *aims to encode input sequence and* L *is a linear layer.* h_{t-1} *is hidden state at time* $t-1$*. We combine* (25) *and* (26) *to get* $\hat{y}_t$ *as follows:*

$$\hat{y}_t = L(F(X_t, h_{t-1})). \tag{27}$$

Proof *Given input* X_t *and hidden state* h_t*, we shift them by* t_0 *respectively to get shifted value of* X_t*,* $X'_t = X_{t-t_0}$ *and shifted value of* h_t*,* $h'_t = h_{t-t_0}$*. It requires that* $\hat{y}_{t-t_0} = \hat{y}'_t$ *if the system is time-invariance. According to* (27)*,* $\hat{y}_{t-t_0}$ *and* $\hat{y}'_t$ *can be expressed as follows, respectively:*

$$\hat{y}_{t-t_0} = L(F(X_{t-t_0}, h_{t-t_0-1})), \tag{28}$$

where $\hat{y}_{t-t_0}$ *is the predicted value corresponding to input* X_{t-t_0}*,* h_{t-t_0-1} *is the hidden state of previous one time step.*

$$\begin{aligned}\hat{y}'_t &= L(F(X'_t, h'_{t-1})) \\ &= L(F(X_{t-t_0}, h_{t-t_0-1})),\end{aligned} \tag{29}$$

where $\hat{y}'_t$ *is the predicted value corresponding to input* X'_t*,* h'_{t-1} *is the hidden state of previous one time step. Therefore, RNN is time-invariant because* $\hat{y}_{t-t_0} = \hat{y}'_t$*.* □

With in-depth discussion and analysis, MATVN breaks the traditional time-invariant sequence modeling setting around RNN, which is beneficial to better capture irregular dynamics in multiple time steps.

3.4 Loss function.

In time series forecasting task, loss function is a measure of the difference between the predicted values of the model $\hat{Y}_{t+1:t+\tau}$ and the ground-truth $Y_{t+1:t+\tau}$. Mean Squared Error (MSE) is a commonly used loss function in time series forecasting. It is worth noting that when using MSE as the loss function, the ultimate goal is to adjust the parameters of the model to make the MSE as small as possible, so as to improve the prediction accuracy of the model. Therefore, our ultimate goal is to minimize the difference between ground-truth $Y_{t+1:t+\tau}$ and forecast $\hat{Y}_{t+1:t+\tau}$ via the following MSE optimization objective loss function:

$$Loss_\Theta = \left\| Y_{t+1:t+\tau} - \hat{Y}_{t+1:t+\tau} \right\|_2^2, \tag{30}$$

where Θ is the learning parameter of network.

4 Experiment

In this section, we begin by describing the real-world time series datasets used in our approach. Then we compare MATVN with classic and state-of-the-art baseline models. Subsequently, we detail some model training settings. Finally, we provide an extensive experimental evaluation and exhibit exhaustive discussions with regard to the forecasting results we have achieved according to our contributions in this paper.

4.1 Data description

Three datasets listed below are used in our paper to evaluate MATVN.

Electricity Transformer Temperature (ETT) contains electricity-related two-year data collected from two Chinese stations. To explore the performance of the model on different granularity data, we use different sampling frequencies to obtain 15-minutes data ETTm1 and hourly data ETTh1, ETTh2. Each data point contains six power load features and the target value oil temperature. The TrainSet/ValSet/TestSet is 12/4/4 months.

Electricity Consuming Load (ECL) datasets collect two-year hourly electricity consumption of 321 clients. Due to missing data, the dataset is converted into hourly consumption for two years, and the target value is set to 'MT-320'. The TrainSet/ValSet/TestSet is 15/3/4 months.

Weather collects the Max-Planck-Institute's climatological data every 10 minutes in 2020 year. Each data point mainly contains 21 climate features including temperature and air pressure etc. The TrainSet/ValSet/TestSet is 28/10/10 months.

4.2 Baseline Methods

We choose the commonly used traditional statistical prediction model and a range of state-of-the-art deep learning methods for time series forecasting as our baselines. According to our proposed innovations, we divide deep learning methods baselines into the following two groups.

To verify the effectiveness of the time-variant mechanism of MATVN, we consider LSTMa, DeepAR, and TCN as our first group of baselines. LSTMa and DeepAR are examples of sequence modeling around RNN-based models. And TCN is another representative of modeling sequence dependencies based on CNN.

We consider Reformer, LogTrans, Informer, and Autoformer as the second group of our baselines. These Transformer-based systems, which are highly effective, however, lack the capacity to enable free window deformation of temporal modeling and capture dependencies at different scales. We aim at demonstrating MATVN's superiority in these two aspects. These baselines are detailed as follows:

- **SARIMA** [6]: SARIMA is a commonly used and one of the most representative methods in the field of traditional time series forecasting.
- **LSTMa** [41]: The attention is added to LSTM for sequence prediction.
- **DeepAR** [42]: DeepAR trains an autoregressive recurrent neural network to generate accurate probabilistic forecasts on the high volume of related time series data.
- **TCN** [43]: TCN employs dilations and casual convolutions to expand the receptive field to better adapt to time series data.
- **Reformer** [44]: Reformer improves the efficiency of Transformer by introducing a locality-sensitive hashing mechanism for self-attention.
- **LogTrans** [23]: LogTrans makes the canonical Transformer sensitive to local context by introducing causal convolution to self-attention.
- **Informer** [45]: Informer alleviates the issues of quadratic memory usage and quadratic time complexity with its proposed distilling operation and ProbSparse self-attention when predicting long-sequence time series.
- **Informer+** [45]: It is an Informer variant in which canonical self-attention is used in place of ProbSparse self-attention.
- **Autoformer** [36]: Autoformer designs a novel decomposition architecture, in which auto-correlation mechanism aims to capture time series dependencies at subsequence level.

4.3 Settings

The ADAM optimizer is used to train our proposed method, with a batch size of 16 and an initial learning rate of 1e-3. Each dataset transmitted into the network is normalized by zero mean. We implement all the experiments with PyTorch [1] on a single NVIDIA TITAN RTX 24 GB GPU. Mean Squared Error (MSE) and Mean Absolute Error (MAE) are used to evaluate our model on three datasets as follows:

$$MSE = \frac{1}{N}\sum_{i=1}^{N}(y_i - \hat{y}_i)^2, \tag{31}$$

$$MAE = \frac{1}{N}\sum_{i=1}^{N}\sqrt{(y_i - \hat{y}_i)^2}, \tag{32}$$

where y_i is ground-truth and $\hat{y}_i$ is prediction and N is the number of testing samples.

4.4 The Performance of Time-Variant Architecture

We repeat each experiment five times, and Tables 1, 2, 3 and 4 summary the evaluation results of MATVN compared to other methods on three real-world datasets. From above tables, the best forecasting results are highlighted in boldface, and the conclusions are drawn in the following paragraphs.

MATVN is superior to these baselines by a large margin, with about 30%-40% improvement in most cases and 60%-

[1] The code has been released at https://github.com/chxiag/MATVN.

Table 1 Univariate forecasting performance (MSE) of MATVN against other baseline models (mean± std)

Methods		SARIMA	DeepAR	LSTMa	Reformer	LogTrans	Informer	Informer+	Autoformer	MATVN
ETTh1	24	0.108	0.107±0.034	0.114±0.023	0.103±0.043	0.022±0.018	0.098±0.023	0.092±0.010	0.033±0.034	**0.018±0.012**
	48	0.175	0.162±0.018	0.193±0.021	0.284±0.022	0.167±0.019	0.158±0.025	0.161±0.018	0.051±0.015	**0.037±0.015**
	168	0.396	0.239±0.023	0.236±0.019	1.522±0.025	0.207±0.028	0.183±0.012	0.187±0.009	0.113±0.022	**0.092±0.025**
	336	0.468	0.445±0.033	0.590±0.023	0.590±0.023	0.230±0.013	0.222±0.024	0.215±0.011	0.107±0.025	**0.104±0.015**
ETTh2	24	3.554	0.098±0.025	0.155±0.028	0.263±0.019	0.102±0.011	0.093±0.022	0.099±0.008	0.114±0.028	**0.088±0.018**
	48	3.190	0.163±0.028	0.190±0.022	0.458±0.012	0.169±0.013	0.155±0.029	0.159±0.018	0.132±0.019	**0.108±0.028**
	168	2.800	0.255±0.020	0.385±0.010	1.029±0.011	0.246±0.019	0.232±0.033	0.235±0.009	0.187±0.035	**0.114±0.018**
	336	2.753	0.604±0.016	0.558±0.018	1.668±0.012	0.267±0.011	0.263±0.015	0.258±0.011	0.246±0.025	**0.118±0.012**
ETTm1	24	0.090	0.091±0.019	0.121±0.018	0.095±0.023	0.065±0.013	0.030±0.025	0.034±0.021	0.019±0.015	**0.005± 0.022**
	48	0.179	0.219±0.019	0.305±0.011	0.249±0.010	0.078±0.009	0.069±0.026	0.066±0.019	0.038±0.016	**0.023±0.012**
	96	0.272	0.364±0.029	0.287±0.022	0.920±0.012	0.199±0.032	0.194±0.021	0.187±0.015	0.056±0.031	**0.027±0.010**
	288	0.462	0.948±0.011	0.524±0.010	1.108±0.009	0.411±0.029	0.401±0.022	0.409±0.007	0.079±0.012	**0.065±0.013**
ECL	48	0.010	0.204±0.013	0.493±0.022	0.971±0.032	0.280±0.011	0.239±0.021	0.238±0.017	0.229±0.031	**0.201±0.011**
	168	1.032	0.315±0.012	0.723±0.014	1.671±0.011	0.454±0.021	0.447±0.017	0.442±0.022	0.417±0.019	**0.308±0.019**
	336	1.136	0.414±0.021	1.212±0.023	0.528±0.027	0.514±0.029	0.489±0.039	0.501±0.012	0.452±0.037	**0.413±0.029**
	720	1.251	0.563±0.027	1.511±0.017	4.891±0.019	0.558±0.029	0.540±0.017	0.543±0.010	0.502±0.018	**0.497±0.028**
weather	24	0.219	0.128±0.016	0.131±0.019	0.231±0.029	0.136±0.028	0.117±0.038	0.119±0.009	X0.109±0.018	**0.103±0.018**
	48	0.273	0.203±0.013	0.190±0.019	0.328±0.008	0.206±0.018	0.178±0.038	0.185± 0.019	0.169±0.018	**0.161±0.018**
	168	0.503	0.293±0.018	0.341±0.011	0.654±0.012	0.309±0.022	0.266±0.012	0.269±0.021	0.249±0.016	**0.217±0.013**
	336	0.728	0.585±0.029	0.456±0.033	1.792±0.031	0.359±0.021	0.297±0.010	0.310±0.013	0.296±0.009	**0.288±0.014**

*Reported metrics except Autoformer and MATVN all come from the Informer paper (Zhou et al., 2021)

Table 2 Univariate forecasting performance (MAE) of MATVN against other baseline models (mean± std)

Methods		SARIMA	DeepAR	LSTMa	Reformer	LogTrans	Informer	Informer+	Autoformer	MATVN
ETTh1	24	0.284	0.280±0.015	0.272±0.026	0.389±0.019	0.259±0.011	0.247±0.010	0.246±0.001	0.152±0.017	**0.113±0.016**
	48	0.424	0.327±0.014	0.358±0.015	0.445±0.008	0.328±0.023	0.319±0.011	0.322±0.012	0.181±0.011	**0.159±0.022**
	168	0.504	0.422±0.013	0.392±0.023	1.191±0.016	0.375±0.009	0.346±0.014	0.355±0.012	0.232±0.015	**0.198±0.024**
	336	0.593	0.552±0.024	0.698±0.016	1.124±0.024	0.398±0.016	0.387±0.014	0.369±0.012	0.258±0.022	**0.219±0.013**
ETTh2	24	0.445	0.263±0.009	0.307±0.015	0.437±0.016	0.255±0.015	0.240±0.014	0.241±0.012	0.249±0.011	**0.214±0.015**
	48	0.474	0.341±0.024	0.348±0.024	0.545±0.025	0.348±0.035	0.314±0.015	0.317±0.012	0.278±0.025	**0.226±0.016**
	168	0.595	0.414±0.018	0.514±0.016	0.879±0.018	0.422±0.022	0.389±0.027	0.390±0.017	0.322±0.019	**0.228±0.025**
	336	0.738	0.607±0.024	0.606±0.018	1.228±0.019	0.437±0.034	0.417±0.033	0.423±0.030	0.389±0.035	**0.243±0.019**
ETTm1	24	0.206	0.243±0.033	0.233±0.022	0.228±0.025	0.202±0.017	0.137±0.018	0.160±0.011	0.132±0.019	**0.052± 0.011**
	48	0.306	0.362±0.026	0.411±0.026	0.390±0.021	0.220±0.019	0.203±0.011	0.194±0.010	0.157±0.010	**0.117±0.011**
	96	0.399	0.496±0.011	0.420±0.010	0.767±0.009	0.386±0.001	0.372±0.009	0.384±0.019	0.183±0.023	**0.129±0.019**
	288	0.558	0.795±0.015	0.584±0.017	1.245±0.019	0.572±0.017	0.554±0.029	0.548±0.011	0.216±0.021	**0.175±0.011**
ECL	48	0.764	0.357±0.011	0.539±0.019	0.884±0.016	0.429±0.011	0.359±0.045	0.368±0.025	0.309±0.044	**0.299±0.018**
	168	0.833	0.436±0.023	0.655±0.020	1.587±0.019	0.529±0.018	0.503±0.017	0.514±0.010	0.458±0.016	**0.433±0.014**
	336	0.876	0.519±0.012	0.898±0.029	2.196±0.028	0.563±0.027	0.528±0.026	0.552±0.016	0.491±0.023	**0.478±0.020**
	720	0.933	0.595±0.018	0.966±0.017	4.047±0.028	0.609±0.025	0.571±0.024	0.578±0.011	0.531±0.021	**0.509±0.021**
weather	24	0.355	0.274±0.021	0.254±0.019	0.401±0.011	0.279±0.012	0.251±0.029	0.256±0.019	0.249±0.019	**0.231±0.010**
	48	0.409	0.353±0.010	0.334±0.011	0.423±0.018	0.356±0.017	0.318±0.016	0.316±0.010	0.314±0.027	**0.306±0.022**
	168	0.599	0.451±0.018	0.448±0.017	0.634±0.019	0.439±0.018	0.398±0.017	0.404±0.009	0.374±0.011	**0.358±0.011**
	336	0.730	0.644±0.011	0.554±0.029	1.093±0.011	0.484±0.039	0.416±0.011	0.422±0.015	0.378±0.018	**0.374±0.025**

Table 3 Multivariate forecasting performance (MSE) of MATVN against other baseline models (mean± std)

Methods		SARIMA	LSTMa	TCN	Reformer	LogTrans	Informer	Informer+	Autoformer	MATVN
ETTh1	24	0.666	0.650±0.022	1.193±0.013	0.991±0.015	0.686±0.019	0.577±0.021	0.620±0.018	0.384±0.024	**0.308±0.029**
	48	0.778	0.702±0.022	1.356±0.014	1.313±0.017	0.766±0.029	0.685±0.011	0.692±0.012	0.392±0.018	**0.314±0.024**
	168	1.245	1.212±0.021	1.907±0.020	1.824±0.018	1.002±0.019	0.931±0.020	0.947±0.014	0.490±0.026	**0.348±0.022**
	336	1.455	1.424±0.019	2.572±0.018	2.117±0.019	1.362±0.022	1.128±0.025	1.094±0.026	0.505±0.023	**0.376±0.019**
ETTh2	24	1.155	1.143±0.021	2.589±0.019	1.531±0.020	0.828±0.032	0.720±0.022	0.753±0.012	0.261±0.019	**0.213±0.011**
	48	1.689	1.671±0.028	2.669±0.019	1.871±0.022	1.806±0.025	1.457±0.011	1.461±0.009	0.312±0.010	**0.247±0.024**
	168	4.169	4.117±0.011	2.875±0.020	4.660±0.013	4.070±0.011	3.489±0.020	3.485±0.010	0.457±0.012	**0.354±0.010**
	336	3.544	3.434±0.010	2.941±0.015	4.028±0.025	3.875±0.019	2.723±0.010	2.626±0.007	0.471±0.011	**0.470±0.011**
ETTm1	24	0.628	0.621±0.025	1.854±0.023	0.724±0.021	0.419±0.020	0.323±0.029	0.306±0.018	0.383±0.020	**0.144±0.019**
	48	1.456	1.392±0.025	1.872±0.023	1.098±0.021	0.507±0.020	0.494±0.019	0.465±0.011	0.454±0.020	**0.159±0.011**
	96	1.445	1.339±0.022	2.669±0.012	1.433±0.025	0.768±0.022	0.678±0.022	0.681±0.010	0.481±0.020	**0.143±0.010**
	288	1.788	1.740±0.024	2.672±0.015	1.820±0.014	1.462±0.019	1.056±0.019	1.162±0.012	0.634±0.011	**0.438±0.025**
ECL	48	0.497	0.486±0.023	0.586±0.028	1.404±0.021	0.355±0.011	0.344±0.015	0.334±0.011	0.286±0.015	**0.046±0.022**
	168	0.654	0.602±0.022	0.682±0.022	1.069±0.023	0.432±0.012	0.424±0.018	0.420±0.013	0.389±0.018	**0.331±0.022**
	336	0.891	0.886±0.015	0.598±0.011	1.601±0.021	0.373±0.022	0.381±0.023	0.381±0.013	0.356±0.020	**0.338±0.021**
	720	1.801	1.676±0.021	0.696±0.012	2.009±0.012	0.409±0.021	0.406±0.021	0.391±0.017	0.387±0.023	**0.377±0.022**
weather	24	1.601	0.546±0.018	0.479±0.012	0.655±0.019	0.435±0.013	0.335±0.017	0.349±0.015	0.249±0.011	**0.235±0.010**
	48	0.867	0.829±0.019	0.486±0.012	0.729±0.011	0.426±0.019	0.395±0.017	0.386±0.009	0.328±0.012	**0.325±0.012**
	168	1.167	1.038±0.012	0.494±0.011	1.318±0.013	0.727±0.011	0.608±0.010	0.613±0.013	0.340±0.023	**0.331±0.023**
	336	1.767	1.657±0.023	0.508±0.012	1.930±0.019	0.754±0.012	0.702±0.013	0.707±0.015	0.359±0.019	**0.342 ± 0.013**

Table 4 Multivariate forecasting performance (MAE) of MATVN against other baseline models (mean± std)

Methods		SARIMA	LSTMa	TCN	Reformer	LogTrans	Informer	Informer+	Autoformer	MATVN
ETTh1	24	0.633	0.624±0.011	0.875±0.012	0.754±0.010	0.604±0.022	0.549±0.018	0.577±0.011	0.425±0.019	**0.405±0.019**
	48	0.688	0.675±0.012	0.930±0.023	0.906±0.031	0.757±0.032	0.625±0.033	0.671±0.022	0.419±0.019	**0.417±0.010**
	168	0.913	0.867±0.020	1.201±0.011	1.138±0.009	0.846±0.001	0.752±0.013	0.797±0.011	0.481±0.034	**0.431±0.033**
	336	0.101	0.994±0.020	1.321±0.011	1.280±0.033	0.952±0.011	0.873±0.010	0.813±0.011	0.484±0.011	**0.439±0.022**
ETTh2	24	0.823	0.813±0.030	1.349±0.011	1.613±0.003	0.750±0.011	0.665±0.029	0.727±0.020	0.341±0.010	**0.338±0.019**
	48	0.132	1.221±0.030	1.578±0.017	1.735±0.019	1.034±0.018	1.001±0.016	1.077±0.011	0.373±0.010	**0.364±0.020**
	168	1.677	1.674±0.019	2.408±0.028	1.846±0.020	1.681±0.011	1.515±0.010	1.612±0.013	0.455±0.018	**0.450±0.019**
	336	1.555	1.549±0.009	2.412±0.018	1.688±0.023	1.763±0.016	1.340±0.019	1.285±0.014	0.475±0.017	**0.472±0.009**
ETTm1	24	0.633	0.629±0.033	1.065±0.023	0.607±0.023	0.412±0.026	0.369±0.019	0.371±0.016	0.403±0.011	**0.236±0.011**
	48	0.992	0.939±0.009	1.105±0.011	0.777±0.021	0.583±0.029	0.503±0.018	0.470±0.010	0.453±0.021	**0.280±0.011**
	96	0.925	0.913±0.022	1.434±0.035	0.945±0.034	0.792±0.046	0.614±0.019	0.612±0.015	0.463±0.023	**0.279±0.010**
	288	1.232	1.124±0.020	1.442±0.011	1.094±0.009	1.320±0.021	0.786±0.010	0.879±0.011	0.528±0.011	**0.354±0.011**
ECL	48	0.581	0.572±0.033	0.672±0.067	0.999±0.011	0.418±0.019	0.393±0.011	0.399±0.013	0.305±0.012	**0.152±0.011**
	168	0.612	0.574±0.010	0.592±0.011	1.515±0.019	0.368±0.011	0.368±0.009	0.353±0.019	0.334±0.019	**0.315±0.011**
	336	0.899	0.886±0.015	0.598±0.023	1.601±0.022	0.373±0.027	0.381±0.026	0.381±0.016	0.356±0.019	**0.338±0.021**
	720	0.899	1.676±0.010	0.696±0.017	2.009±0.018	0.409±0.020	0.406±0.021	0.391±0.011	0.387±0.023	**0.377±0.029**
weather	24	0.615	0.546±0.022	0.479±0.010	0.655±0.019	0.435±0.022	0.335±0.012	0.349±0.007	0.249±0.011	**0.235±0.022**
	48	0.861	0.829±0.010	0.486±0.011	0.729±0.013	0.426±0.011	0.395±0.016	0.386±0.012	0.328±0.013	**0.325±0.015**
	168	1.112	1.038±0.010	0.494±0.038	1.318±0.032	0.727±0.019	0.608±0.014	0.613±0.004	0.340±0.011	**0.331±0.011**
	336	1.677	1.657±0.010	0.508±0.022	1.930±0.030	0.754±0.010	0.702±0.033	0.707±0.013	0.359±0.011	**0.342±0.022**

70% improvement in some cases and its forecasting errors rise steadily rather than sharply within the growing number of prediction steps, which demonstrates that MATVN has excellent sequence modeling ability in multi-step time series forecasting problems.

From above tables, it is evident that SARIMA does not exhibit superior prediction performance in long-term forecasting compared to other deep learning methods. This is mainly because that SARIMA model is still based on the assumption of linearity, which assumes that the seasonality and trend of the time series are linear. This may not be sufficient to capture some nonlinear relationships, especially when there are complex nonlinear patterns in the data, such as ETTm1 dataset.

For LSTMa and DeepAR, MATVN outperforms these RNN-based neural networks in both multivariate and univariate settings. The MATVN model, in particular, exhibits outstanding performance in the following scenarios. For univariate prediction results, MATVN outperforms LSTMa across ETTh1, ETTh2, ETTm1 and Weather datasets on MSE by decreasing 61.16% in average when prediction length is 24. And MATVN outperforms LSTMa across ETTh1, ETTh2, ECL and Weather datasets on MAE by decreasing 51.94% in average when prediction length is 336. Furthermore, when comparing the performance of MATVN and LSTMa on MSE and MAE metrics for multivariate prediction across the ETTh1, ETTh2, ECL, and Weather datasets, MATVN exhibits significant superiority. It achieves an average decrease of 68.95% on MSE when prediction length is 168 and 66.64% on MAE when prediction length is 336. We speculate that this is mainly because these RNN-based neural networks use parameter sharing by repeating a set of fixed architectures with fixed parameters in time or space, which limits the ability of RNN-based methods to capture irregular and nonlinear time series data. Fortunately, compared with time-invariant RNN-based models, MATVN allows the parameters of the model to adjust accordingly with the irregular changes of these scales, thus capturing the sharp changes of the sequence more sensitively. From a global perspective, LSTMa and DeepAR exhibit worse performance on longer forecast steps for almost all datasets. It can be confidently surmised that although they are designed to perform better relative to simple RNNs, they still suffer from the problem of gradient vanishing, especially when dealing with very long sequences. In these cases, it is still possible for the gradient to gradually decrease, leading to difficulty in convergence or poor training results.

The other most typical paradigm in temporal modeling is TCN, which mainly capitalizes on dilated convolutions for expanding the receptive field on input data to support sequence prediction. Similar to RNN-based architectures, CNN-based models achieve shift invariance in the spatial domain, which they translate into time-invariance in time series applications. We conjecture that time invariance prevents CNN-based models from sensitively capturing changes in irregular sequences, thus reducing the ability to model complex dependencies and hindering the capacity to perform multi-step-ahead time series forecasting. Based on our analysis, we attribute the following experiment results to this underlying cause. MATVN outperforms TCN across ETTh1, ETTh2, ECL and Weather datasets, the MSE decreases 61.35% in average when prediction length is 336. And MATVN also demonstrates better performance than TCN on the ETTh1, ETTh2, ETTm1, ECL and Weather datasets, with an average MAE reduction of 67.61% when prediction length is 48.

4.5 The performance of multi-scale adaptive sequence modeling

We conduct a series of experiments on these datasets to analyze which of the following scale-aware neural networks listed below can better improve inference performance.

- MATVN-FS: Its hidden module is equipped with the SMF module, which aims to capture single-scale fixed-scope local dependencies in the sequence.
- MATVN-VM: It replaces the SMF module with the MMD module to learn multi-scale fixed-scope time series information.
- SATVNN: Its hidden module makes the contribution of tokens with different distances to current token obeys Cauchy distribution. This attenuation mechanism makes the model more flexible to capture single-scale local correlation of time series data.
- MATVN: It replaces the MMD module with the MMA module to flexibly capture multi-scale temporal dynamics.

The comparison experimental results of four models on all benchmark datasets are visualized in Fig. 5. It is apparent that MATVN achieves the lowest error across all datasets, and we also draw the following conclusions. First, the capacity of model to learn sequential dependencies is enhanced when the number of windows (scales) is increased from 1 to 3 under the same fixed-scope or more flexible sequence modeling way. In particular, MATVN-VM model's MSE on the ETT* datasets is 4.24% lower than that of MATVN-FS in average. On the ECL dataset, the MAE of MATVN-VM is 7.60% lower than that of MATVN-FS, and the MAE obtained by MATVN on Weather datasets is also 5.34% lower than that of SATVNN. This is mainly attributed to more diverse dynamics at different temporal resolutions can be captured by MATVN-VM and MATVN than MATVN-FS and SATVNN.

Subsequently, we find that when we modify the fixed-scope modeling way to the adaptive modeling way, the

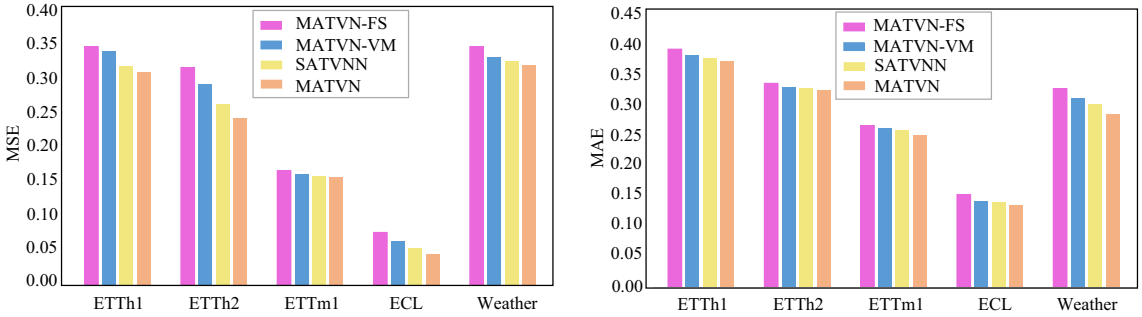

Fig. 5 Multivariate forecasting results with a prediction length of 48 for MATVN-FS, MATVN-VM, SATVNN and MATVN on all real-world datasets

forecasting results are greatly improved in both single-scale modeling and multi-scale modeling. More specifically, on the ETTh2 dataset, the MSE of SATVNN is 16.82% lower than that of MATVN-FS, and the MSE obtained by MATVN is 16.55% lower than that of MATVN-VM. This is mainly due to the fact that adaptive scale modeling allows the model flexibility to choose appropriate attention scopes according to the requirements of the task and the data, thus capturing the relationships between tokens more accurately. This flexibility improves the model's sensitivity to the data and contributes to better modeling. Moreover, in order to demonstrate the stronger generalization ability of the proposed model, we further elaborate on the superiority of MATVN against other state-of-the-art Transformer-based methods by comparing prediction performance under multiple future horizons including {24, 48, 96, 168, 288, 336, 720}, respectively. This design precisely meets the definition of multi-step time series forecasting. The following results are univariate and multivariate settings.

Univariate results In this part, we elaborate on forecasting performance of MATVN against other Transformer-based models. It is apparent from Tables 1 and 2 that, compared with extensive baselines, MATVN still significantly enhances the inference capacity across all datasets. In particular, MATVN achieves 10.83% (0.240 → 0.214) improvement on MAE and 5.37% (0.093 → 0.088) improvement on MSE on ETTh2 against Informer when prediction length is 24. For ETTh1 and ETTm1 dataset, MATVN surpasses Autoformer by 43.13% on MAE in average when prediction length is 24, 33.46% on MSE in average when prediction length is 48. MATVN also beats other baselines on ECL and Weather datasets. We credit MATVN's remarkable inference performance regardless of prediction length to the adaptive window setting, which increases the network's flexibility in adapting to and adequately capturing time series dependencies.

Multivariate results Of note, our proposed MATVN can easily transform univariate time series forecasting into a multivariate one by adjusting the last layer. With regard to the multivariate setting, MATVN exhibits consistent strong performance on all benchmark datasets and all prediction length settings, as shown in Tables 3 and 4. For example, MATVN yields 30.37% averaged relative MSE reduction on ETT* datasets when prediction length is 24 and makes 35.23% averaged relative MSE reduction on ETT* datasets when prediction length is 48 compared to suboptimal results. On the ECL and Weather datasets, MATVN makes 30.27% MAE reduction in average when prediction length is 48, 11.50% MAE reduction in average when prediction length is 168, and 4.89% MSE reduction in average when prediction length is 336. From the above experimental results, we can analyze that compared to the single-scale sequence modeling way or fixed-scale sequence modeling way, the multi-scale adaptive sequence modeling way allows the model to better adapt to the multi-scale variations of the data, providing more accurate, comprehensive and flexible analysis and prediction results. This is mainly due to the fact that multi-scale adaptive sequence modeling way allows the model to adjust and choose the appropriate scale according to the different characteristics of the data and the needs of the problem. Making the proposed algorithm highly flexible in data analysis and model construction.

4.5.1 Complexity analysis

Our current implementation of MMA involves introducing a mask matrix to the canonical MSA. As we all know, a time series tensor $X_{1:t} = [X_1, X_2, \cdots, X_t] \in \mathbb{R}^{B\times L\times d_{in}}$ is first linearly converted to Q, K and V, resulting in $3Ld_{in}^2$ computations. Subsequently, as shown in (2) and (3), matrix multiplication between Q and K yields computational complexity $L^2 d_{in}$, further followed by multiplying the outcome with matrix V to achieve computational complexity $L^2 d_{in}$. Therefore, the overall complexity of canonical MSA amounts to $3Ld_{in}^2+2L^2 d_{in}$. As shown in (14), the extra computations caused by MMA stem from incorporation of the mask matrix. As a result, the model's complexity transforms into $3Ld_{in}^2+(2d_{in}+1)L^2$.

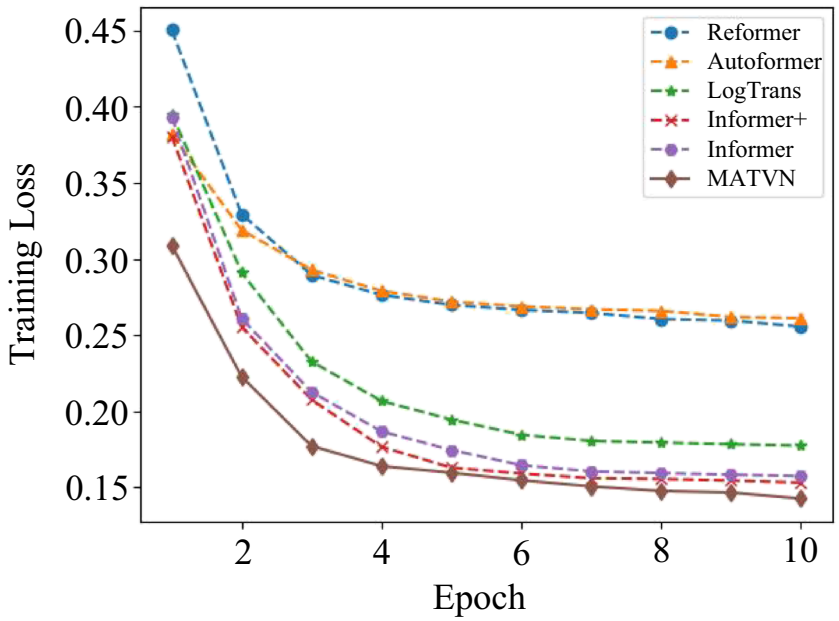

Fig. 6 Comparison of training loss curves between MATVN and other Transformer-based models on ETTh1 datasets

Drawing upon the above derivation, the newly proposed MMA does not increase the complexity greatly compared with the canonical MSA. But it's worth noting that MMA is able to adjust fixed-size attention scope to an adaptive-size proper one for each token in time series. Well-suited window size can preserve crucial local structures, leading to enhanced efficiency in feature learning. As a result, MMA makes MATVN achieve faster learning speed. We draw the training loss curves for both MATVN and other Transformer-based baseline models in Fig. 6. It is observed that during the first few epochs, the training loss of MATVN exhibits faster convergence and lower training error. The comparisons of the number of free parameters and computational cost are also summarized in Table 5. From it we can see that MATVN has the lowest number of free parameters and computational cost.

4.6 Ablation studies

In this section, we further conduct ablation studies and investigate the effectiveness of key components of MATVN.

Is Time-Variant architecture necessary? To emphasize the ability of Time-variant model to adapt to dynamic changes in time series better than Time-invariant model, we plot some forecasts of MATVN and LSTMa in Fig. 7. It illustrates that MATVN performs better than LSTMa in predicting wave peaks and troughs at the beginning. In addition, we found that MATVN is also capable of making very accurate long-term prediction. It can capture irregular inflection points more effectively than LSTMa can. To compare with LSTMa, our model offers better prediction in the central region, and prediction series is more fluid. These findings demonstrate that MATVN is capable of capturing the irregular dynamic changes in real-world sequences and making accurate prediction.

What is the effect of the number of encoder layers of MMA module on the prediction? Stacking multiple encoder layers can improve prediction performance. The line plotted in Fig. 8 shows that the model's performance improves significantly as the number of encoding layers increases. Excessive encoder layers, however, will result in over-fitting. In our experiment, the forecasting performance of MATVN is best with two encoder layers when the length of the input is less than 96, and its forecasting performance reaches the best with three encoder layers when the input length is greater than 96.

How does the feature extractor E() affect results? The sub-network can learn the offset for each token. In contrast

Table 5 Comparison of number of free parameters and computational cost between our proposed model and other Transformer-based models

Methods	Reformer	LogTrans	Informer	Informer+	Autoformer	MATVN
Free Parameters	8683.52k	10045.23k	11328.00k	11330.055k	10535.94k	8626.44k
Computational Cost	20002.11M	19532.21M	17404.53M	17435.26M	29516.39M	356.92M

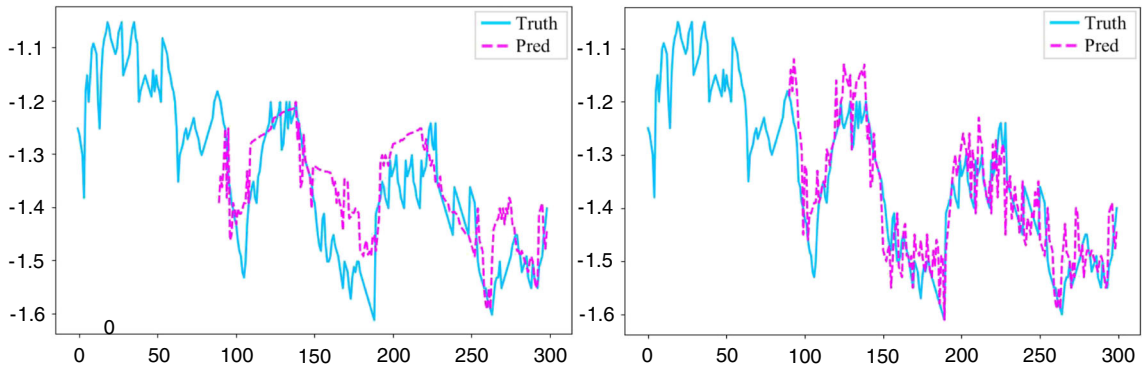

Fig. 7 The long-term prediction performance of LSTMa and MATVN on the ETTm1 dataset

to the fixed-window of each token, the offset is predicted to shift the fixed-window towards more important regions so that better maintaining local structures, which facilitates our model to perform prediction. When the offset is not predicted, the window assigned to each token is fixed, which is the same as the multi-scale self-attention mechanism. Table 6 illustrates how this offset factor influences our model, and from it we can see that predicting offsets can improve 6.799% (MAE) and 5.836% (MSE) averagely above fixed-window settings, which verifies that offsets are critical for our model.

How is the offset generated? As we have stated, the offset is generated by a sub-network that consumes the query features and outputs the offset values of the fixed-window. As depicted in Fig. 9, we implement the sub-network as two linear layers with a nonlinear activation. The input features are first passed through two linear layers to capture features (d_m is the transformed dimension after the first linear layer). Then, a sigmoid activation is adopted to get the offsets.

What WindowSize and Group should be configured? In Tables 7 and 8, we conduct experiments to investigate the effects of WindowSize and Group on the prediction performance of three datasets. Performance is initially improved by increasing the WindowSize win_1, but further increasing it leads to performance degradation. To analyze this result, we consider the extreme case in which win_1 equals to L, excessive global noise will be introduced. In the opposite extreme, if win_1 is set extremely small, the model loses sensitivity to local dependencies. After determining win_1, we set $win_2 = 2win_1$, $win_3 = 3win_1$,..., in our design. The more groups, the more scales we can capture. Our selection, i.e., Group = 3, is the most robust strategy considering performance and time efficiency.

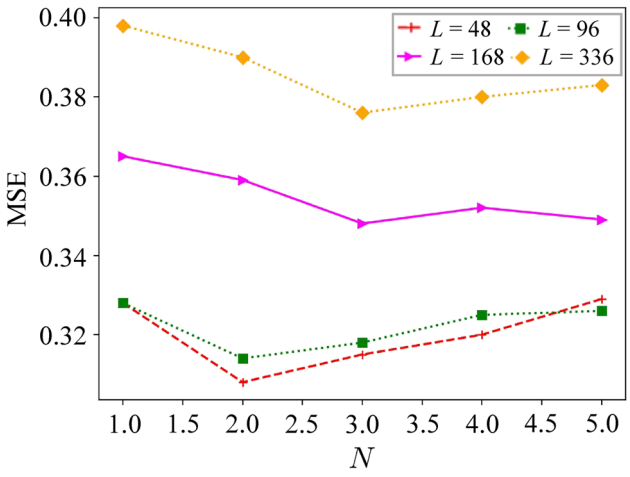

Fig. 8 The effect of the number of encoder layers on Multivariate prediction performance of our MATVN on the ETTh1 dataset with input length of {48, 96,168,336}, respectively

5 Conclusion

In this research, we propose a novel methodology named MATVN. It breaks the traditional time-invariant modeling way, such as CNNs-based and RNNs-based approaches. Because of the design of its time-variant architecture, MATVN can capture irregular fluctuations more sensitively and achieve outstanding performance compared to the above two types of approaches. Based on the time-variant capability-building of MATVN, we further capitalize on proposed MMD module for extracting time series dependencies

Table 6 The effect of offsets on Multivariate prediction performance of our MATVN on three datasets with input length of 48

Offsets			✓	✓		
					Improvement(%)	
	MAE	MSE	MAE	MSE	MAE	MSE
ETTh1	0.455	0.337	**0.417**	**0.314**	8.351 ↑	6.824 ↑
ETTh2	0.387	0.265	**0.364**	**0.247**	5.943 ↑	6.792 ↑
ETTm1	0.296	0.167	**0.280**	**0.159**	5.405 ↑	4.790 ↑
ECL	0.168	0.048	**0.152**	**0.046**	9.523 ↑	4.166 ↑
weather	0.335	0.348	**0.319**	**0.325**	4.776 ↑	6.609 ↑

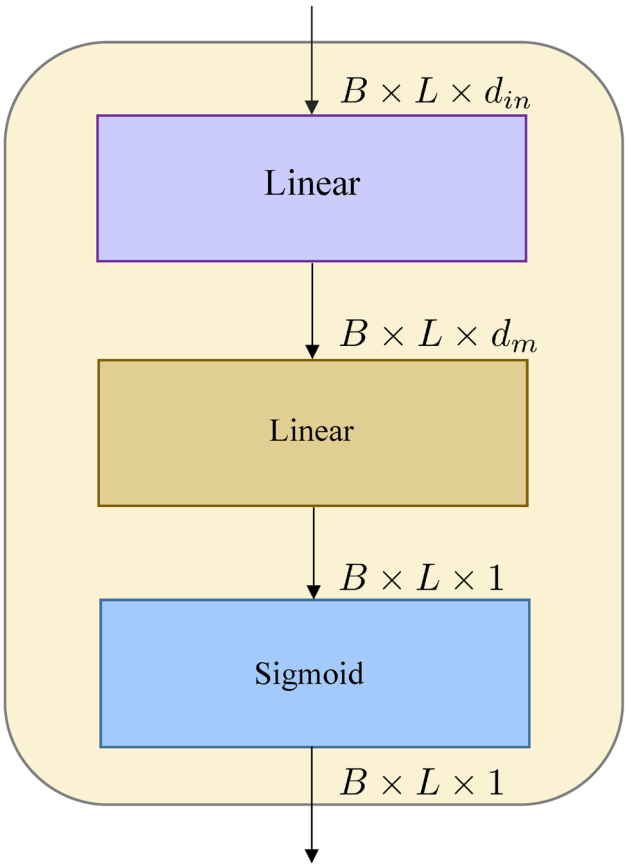

Fig. 9 It illustrates the structure of offset generation network

from different pre-defined scale ranges to boost the model's ability to mine multi-scale information of time series from multiple perspectives. Finally, we design a MMA module, which introduces a new Adaptive Window-aware Mask strategy into MMD module to improve the ability of our proposed model to adaptively learn different attention range representations for different tokens in time series, and successfully enhance the flexibility of the model. We demonstrate through a series of experiments that the proposed method delivers significant performance gains over classical techniques and achieves strong competitiveness when compared to state-of-the-art deep learning models. However, we think that the offset generation network in MMA should be improved. Specifically, the architecture of offset generation network is based on a simple two-layers feed-forward neural network, which lacks flexibility. Therefore, in the future, we should skillfully construct and optimize the offset generation network so that we can better predict the precise window offset for each token in the sequence to better retain the multi-scale local features and provide more accurate prediction results.

Table 7 The effect of different WindowSizes on Univariate prediction performance of our MATVN on three datasets in terms of MSE with input length of 48

WindowSize (win_1)	ETTh1	ETTh2	ETTm1	ECL	Weather
L / 12	0.041	0.113	0.027	0.205	0.167
L / 8	0.039	0.109	0.025	0.203	0.162
L / 6	**0.037**	**0.108**	**0.023**	**0.201**	**0.161**
L / 4	0.042	0.112	0.026	0.206	0.163
L / 3	0.045	0.114	0.028	0.207	0.166

Table 8 The effect of different Groups on Multivariate prediction performance of our MATVN on three datasets in terms of MAE with input length of 48

Group	ETTh1	ETTh2	ETTm1	ECL	Weather
1	0.427	0.369	0.291	0.159	0.322
2	0.423	0.366	0.285	0.156	0.320
3	**0.417**	**0.364**	**0.280**	**0.152**	**0.319**
4	0.419	0.367	0.281	0.154	0.325
5	0.421	0.372	0.283	0.155	0.326

Acknowledgements This research was supported by National Key R & D Program for the Core Technology of Innovation and Entrepreneurship based on AI under Grant 2019YFB1405202.

Data Availability We use real-world datasets that are collected by [45].

Declarations

Conflicts of interest The authors have no relevant financial or non financial interests to disclose.

References

1. Xu C, Zhang A, Xu C, Chen Y (2022) Traffic speed prediction: spatiotemporal convolution network based on long-term, short-term and spatial features. Appl Intell 52(2):2224–2242
2. Banerjee S, Lian Y (2022) Data driven covid-19 spread prediction based on mobility and mask mandate information. Appl Intell 52(2):1969–1978
3. Peng Y, Gong D, Deng C, Li H, Cai H, Zhang H (2022) An automatic hyperparameter optimization DNN model for precipitation prediction. Appl Intell 52(3):2703–2719
4. Taieb S, Bontempi G, Atiya A, Sorjamaa A (2011) A review and comparison of strategies for multi-step ahead time series forecasting based on the nn5 forecasting competition. Expert Syst Appl 39(8):7067–7083
5. Box G, Jenkins G (1976) Time series analysis: Forecasting and control (2nd ed)
6. Guin A (2006) Travel time prediction using a seasonal autoregressive integrated moving average time series model. In: IEEE Intelligent Transportation Systems Conference 493–498
7. Nawaz M, FournierViger P, Shojaee A, Fujita H (2021) Using artificial intelligence techniques for COVID-19 genome analysis. Appl Intell 51(5):3086–3103
8. Rumelhart D, Hinton G, Williams R (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536

9. Acharya U, Fujita H, Oh S, Hagiwara Y, Tan J, Adam M, Tan R (2019) Deep convolutional neural network for the automated diagnosis of congestive heart failure using ECG signals. Appl Intell 49(1):16–27
10. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Polosukhin I (2017) Attention is all you need. In: Advances in neural information processing systems 6000–6010
11. Dabrowski J, Zhang Y, Rahman A (2020) A time-variant deep feed-forward neural network architecture for multi-step-ahead time series forecasting. In: International Conference on Neural Information Processing 579–591
12. Oh J, Wang J, Tang S, Sjoding M, Wiens J (2019) Relaxed parameter: sharing Effectively modeling time-varying relationships in clinical time-series. PMLR 27–52
13. Kag A, Saligrama V (2021) Training recurrent neural networks via forward propagation through time. In: International Conference on Machine Learning 139:5189–5200
14. Yue X, Zhang C, Fujita H, Lv Y (2021) Clothing fashion style recognition with design issue graph. Appl Intell 51(6):3548–3560
15. LeCun, Y., Bengio, Y.: Convolutional networks for images, speech, and time series. The handbook of brain theory and neural networks 3361(10) (1995)
16. Gao C, Zhang N, Li Y, Bian F, Wan H (2022) Self-attention-based time-variant neural networks for multi-step time series forecasting. Neural Comput Appl 34(11):8737–8754
17. Michael, M.: Induction of multiscale temporal structure. In: Advances in neural information processing systems, 275–282 (1991)
18. Wang M, Deng Z (2018) Densely connected cnn with multi scale feature attention for text classification. In: International Joint Conference on Artificial Intelligence 4468–4474
19. Zhang Z, Zhao T, Gay H, Zhang W, Sun B (2021) Weaving attention u-net: A novel hybrid CNN and attention-based method for organs-at-risk segmentation in head and neck CT images. Med Phys 48(11):7052–7062
20. Ma Q, Yan J, Lin Z, Yu L, Chen Z (2021) Deformable self-attention for text classification. Trans Audio Speech Lang Process 29:1570–1581
21. Guo Q, Qiu X, Liu P, Xue X, Zhang Z (2020) Multi scale self-attention for text classification. In: Association for the Advancement of Artificial Intelligence 34(5):7847–7854
22. Shen T, Zhou T, Long G, Jiang J, Pan S, Zhang C (2018) Directional self-attention network for rnn/cnn-free language understanding. In: Association for the Advancement of Artificial Intelligence 32(1):5446–5455
23. Li S, Jin X, Xuan Y, Zhou X, Chen W, Wang Y, Yan X (2019) Enhancing the locality and breaking the memory bottleneck of transformer on time series forecasting. In: Neural Information Processing Systems 32:5244–5254
24. Li Y, Zhang K, Cao J, Timofte R, Van GL (2021) Localvit: Bringing locality to vision transformers. In: Conference on Computer Vision and Pattern Recognition. arXiv:2104.05707
25. Hajirahimi Z, Khashei M (2023) Hybridization of hybrid structures for time series forecasting: a review. Artif Intell Rev 56(2):1201–1261
26. Wu Y, Zhao X, Li Y, Guo L, Zhu X, Fournier-Viger P, Wu X (2022) OPR-Miner: Order-preserving rule mining for time series. arXiv:2209.08932
27. Chen H, Rossi RA, Mahadik K, Kim S, Eldardiry H (2023) Graph deep factors for probabilistic time-series forecasting. ACM Trans Knowl Discov Data 17(2):26–12630
28. Ilhan F, Karaahmetoglu O, Balaban I, Kozat S (2021) Markovian rnn: An adaptive time series prediction network with hmm-based switching for nonstationary environments. IEEE Trans Neural Netw Learn Syst, 1–14
29. Cirstea R, Kieu T, Guo C, Yang B, Pan S (2021) Enhancenet: Plugin neural networks for enhancing correlated time series forecasting. In: International Conference on Data Engineering 1739–1750
30. Bai S, Kolter J, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271
31. Liu S, Ji H, Wang M (2020) Nonpooling convolutional neural network forecasting for seasonal time series with trends. IEEE Trans Neural Netw Learn Syst 31(8):2879–2888
32. Tang W, Long G, Liu L, Zhou T, Blumenstein M (2022) Omni-scale cnns: a simple and effective kernel size configuration for time series classification. In: International Conference on Learning Representations
33. Choromanski K, Likhosherstov V, Dohan D, Song X, Gane A, Sarlos T, Hawkins P, Davis J, Mohiuddin A, Kaiser L, Belanger D, Colwell L, Weller A (2021) Rethinking attention with performers. In: International Conference on Learning Representations
34. Fan Z, Liu Z, Wang A, Nazari Z, Zheng L, Peng H, Yu P (2022) Sequential recommendation via stochastic self-attention. In: Proceedings of the ACM Web Conference 2036–2047
35. Lin Y, Koprinska I, Rana M (2021) Ssdnet: State space decomposition neural network for time series forecasting. In: International Conference on Data Mining 370–378
36. Wu H, Xu J, Wang J, Long M (2021) Autoformer: Decomposition transformers with auto-correlation for long-term series forecasting. In: Advances in Neural Information Processing Systems 34:22419–22430
37. Wu N, Green B, Ben X, Banion S (2020) Deep transformer models for time series forecasting: The influenza prevalence case. arXiv:2001.08317
38. Zhou T, Ma Z, Wen Q, Wang X, Sun L, Jin R (2020) Fedformer: Frequency enhanced decomposed transformer for long-term series forecasting. In: International Conference on Machine Learning. arXiv:2201.12740
39. Shih SH, Tsokos CP (2008) A weighted moving average process for forecasting. J Mod Appl Stat Meth 7(1):15
40. Oppenheim A, Schafer R, Buck J (2009) Pearson education signal processing series. Discrete-time signal processing (2nd ed)
41. Bahdanau K, Cho D, Bengio Y (2015) Neural machine translation by jointly learning to align and translate. In: International Conference on Learning Representations.arXiv:1409.0473
42. Salinas D, Flunkert V, Gasthaus J, Januschowski T (2022) Deepar: Probabilistic forecasting with autoregressive recurrent networks. Int J Forecasting 36(3):1181–1191
43. Bai S, Kolter J, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. arXiv:1803.01271
44. Liu S, Ji H, Wang M (2020) Reformer: The efficient transformer. In: International Conference on Learning Representations. arXiv:2001.04451
45. Zhou H, Zhang S, Peng J, Zhang S, Li J, Xiong H, Zhang W (2021) Informer: Beyond efficient transformer for long sequence time-series forecasting. In: Association for the Advancement of Artificial Intelligence 35(12):11106–11115