Neural Computing and Applications (2022) 34:8737–8754
https://doi.org/10.1007/s00521-021-06871-1

ORIGINAL ARTICLE

Self-attention-based time-variant neural networks for multi-step time series forecasting

Changxia Gao[1,2] · Ning Zhang[1,2] · Youru Li[1,5] · Feng Bian[3] · Huaiyu Wan[1,4]

Received: 10 February 2021 / Accepted: 12 December 2021 / Published online: 5 February 2022

Abstract
Time series forecasting is ubiquitous in various scientific and industrial domains. Powered by recurrent and convolutional and self-attention mechanism, deep learning exhibits high efficacy in time series forecasting. However, the existing forecasting methods are suffering some limitations. For example, recurrent neural networks are limited by the gradient vanishing problem, convolutional neural networks cost more parameters, and self-attention has a defect in capturing local dependencies. What's more, they all rely on time invariant or stationary since they leverage parameter sharing by repeating a set of fixed architectures with fixed parameters over time or space. To address the above issues, in this paper we propose a novel time-variant framework named Self-Attention-based Time-Variant Neural Networks (SATVNN), generally capable of capturing dynamic changes of time series on different scales more accurately with its time-variant structure and consisting of self-attention blocks that seek to better capture the dynamic changes of recent data, with the help of Gaussian distribution, Laplace distribution and a novel Cauchy distribution, respectively. SATVNN obviously outperforms the classical time series prediction methods and the state-of-the-art deep learning models on lots of widely used real-world datasets.

Keywords Multi-step time series forecasting · Self-attention · Time variant · Recent changes in data · Different scales changes in data

1 Introduction

Time series forecasting is recognized as establishing an appropriate prediction model for sequences arranged in chronological order to make the most of complex sequential dependencies [1]. To date, a considerable amount of literature has been published to solve this prediction problem that has attracted much attention. As representative of statistical regression methods, auto-regression (AR) model and its variant auto-regressive moving average (ARMA) model which is composed of an auto-regressive model and a moving average model are well known for time series forecasting. However, the disadvantage of AR and ARMA approach referenced above and another forecasting method [2] is that it is difficult for them to capture the nonlinear dynamics of time series. Machine learning methods are a major solution in the field of time series forecasting. Many researchers are committed to developing nonlinear models in a variety of ways to address time series forecasting problem such as time series prediction model based on kernel method [3], ensemble method [4] and Gaussian processes [5]; however, these methods may not properly capture the real potential nonlinear relationships due to employing a predefined nonlinear form.

Recently, there is a growing body of literature that recognizes the importance of addressing the issue of time

✉ Huaiyu Wan
hywan@bjtu.edu.cn

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

2 China Engineering Research Center of Network Management Technology for High Speed Railway of MOE, Beijing 100044, China

3 College of Information Science and Technology, Beijing Normal University, Beijing 100875, China

4 Beijing Key Laboratory of Traffic Data Analysis and Mining, Beijing 100044, China

5 Beijing Key Laboratory of Advanced Information Science and Network Technology, Beijing, China

series forecasting with deep learning methods. A recurrent neural network (RNN) is famous for sequence modeling due to its ability in capturing nonlinear relationships. Proceeding from significant analysis of [6] and a great many researches, it can be concluded that RNN is easy to cause a vanishing gradient problem. Long short-term memory neural network (LSTM) is of great significance because it makes the first attempt to alleviate vanishing gradient problem in RNN [7–9]. However, the issue still remains unresolved. As an example, [10] indicated that the effective context size of the language model using LSTM was about 200 tokens on average, but can only clearly distinguish 50 tokens nearby, showing that even LSTM was difficult to capture long-term dependencies. Inspired by some successful applications appeared in machine translation, the overwhelming majority of researches commented that sequence-to-sequence (Seq2Seq) model [11] effectively encoded the input information, which was conductive to time series forecasting.

More recently, there has been a surge of interest in framework based solely on attention mechanisms [12]. Its architecture is similar to Seq2Seq, but far superior to Seq2Seq in terms of parallel computing and long-term dependency modeling. The work proposed by [12] concluded that the attention mechanisms effectively modeled dependencies of sequences without being affected by distance, which was particularly suitable for capturing recurring patterns. However, self-attention has a weakness in dispersing the distribution of attention because all the signals are considered simultaneously by weighted averaging, which hinders its ability to capture local dependencies of sequences. To address this issue, [13] presented a Gaussian Transformer model for natural language inference to effectively capture the importance of words in the context and achieved competitive performance on the SNLI dataset. Despite the fact that Gaussian prior distribution effectively enhances the impact of adjacent words, the importance of non-adjacent words rapidly tends to zero, which limits the contribution of the important non-adjacent words to the current word representation. The prior distribution of the distance-based self-attention network proposed by [14] is essentially a Laplace prior distribution. The success of Laplace prior distribution lies in the unlimited expression of non-adjacent words; however, the importance of adjacent words is not prominent enough. Similarly, as far as time series forecasting is concerned, the historical data of the time series will have different reference values for future forecasts due to different degrees of old and new data. In order to improve the prediction accuracy, we hope to utilize different weights to measure the reference value of historical data. The newer the data, the higher the reference values and the higher the weights. Therefore, in addition to natural language inference, a variant self-attention mechanism that can capture local dependencies is also applicable to time series tasks.

Recent advances in time-variant methods have facilitated the investigation of time series forecasting problem. A recently published article [15] threw doubt on the original structure of RNN. This study set out to shine new light on time-variant debates through an examination of the time invariant. Time-variant neural networks overcome the shortcoming of network parameters being used repeatedly at each time step and make the prediction more accurate. However, time-variant neural networks begin to expose their defects in capturing the recent trend of nonlinear time series.

To address these challenges, we propose a Self-Attention-based Time-Variant Neural Network (SATVNN) which utilizes its overall framework to provide a time-variant model. Additionally, interleaved outputs of SAVNN assist in mitigating vanishing gradients. What is more, a self-attention block as SATVNN' score component is designed to capture local dependencies of temporal data, in which Gaussian self-attention, Laplace self-attention and a novel Cauchy self-attention are, respectively, proposed to enable the proposed framework to capture recent changes in temporal data.

In summary, the key contributions of this paper are as follows:

- We investigate the challenges of time series forecasting and propose a Self-Attention-based Time-Variant Neural Network (SATVNN) framework. Compared with the existing time series prediction models, the proposed framework can capture the dynamic changes of time series on different scales, especially in capturing the dynamic changes of the recent data.
- A novel time-variant structure is constructed to better learn the dynamics at different scales that span multiple time steps and to mitigate vanishing gradient problem.
- A novel Self-Attention block with three different distributions is designed to reflect recent changes in temporal data, among which a novel Cauchy self-attention mechanism achieves better results than the Gaussian self-attention and the Laplace self-attention.
- Extensive experiments are conducted on some widely used complex seasonal time-series datasets to evaluate our proposed SATVNN framework. The results indicate that our method yields higher prediction accuracy compared with other state-of-the-art methods.

The remainder of this paper is organized as follows. Section 2 reviews some related works. Section 3 provides the details of our SATVNN framework employed in the study. Section 4 presents the experimental evaluations. Section 5 concludes the paper with the way forward.

2 Related work

Multi-step time series forecasting is in sharp contrast with one-step-ahead forecasting [16–18]. It often suffers from the cumulative errors [19]. Recently, a considerable amount of methods based on RNNs and their variants have been proposed to improve the prediction results of multi-step time series. [20] introduced a framework named DeepAR, which used an auto-regressive recurrent network architecture to produce probabilistic forecasts. In contrast, [21] leveraged RNN to map the features to the parameters of the state-space model, which then devoted to predicting the probability distribution of the values at each time step in the sequence. [22] divided the factors that affect the prediction results into random effect and fixed effect, estimated the two factors using different models and then coupled them together to get the final predicted values. Notwithstanding, it is well known that due to the recursive nature of RNNs, the problem of vanishing gradients is prone to occur, which makes it difficult for RNNs to capture long temporal patterns [23].

Seq2Seq has been used in many investigations and studies to capture long-term dependencies [24]. [25] proposed a model for the efficient learning of nonlinear dynamics using a higher-order tensor RNN. [26] presented a multi-horizon quantile prediction model based on the Seq2Seq framework, which is dedicated to solving the regression problem of large-scale time series. [27] presented a loss function called shape and time distortion and combined with Seq2Seq to solve the problem of multi-step prediction on non-stationary signals. [28] built a model to predict extreme event. The main idea of this model was to divide the uncertainty into model uncertainty and forecast uncertainty. It is worth noting that Transformer has received increased attention across many disciplines in recent years. Its architecture is similar to Seq2Seq, but far outperforms Seq2Seq in term of parallel computing and long-term dependencies modeling.

Transformer has been widely used in many fields such as neural machine translation [29–32], natural language inference [33–35], recommendation [36–39], abstractive summarization [40–43] and so on. Due to its excellent performance, Transformer has led to the successful application for time series forecasting. [44] used the Transformer to predict influenza-like illness (ILI). [45] designed a unique structure with a dual self-attention network for forecasting multivariate time series. [46] designed a model called SAnD, the main component of which is a self-attention mechanism. The SAnD model also employed dense interpolation and positional encoding to incorporate temporal orders. We notice that the proposal of the Transformer algorithm inspires a new idea for time series prediction, but there is still room for improvement in capturing local dependencies [47].

There is another tendency for academics to devote some focus to the time-variant models. [15] utilized a feed-forward neural network for multi-step time series forecasting. The novelty is that its architecture breaks the traditional modeling architectures such as RNN or CNN. Specifically, its parameters and structure change over time, which makes the model not time invariant. In spite of the unique structure of the time-variant model being proven to be effective, its ability to capture recent trends in complex time series is insufficient.

At the best of our knowledge, it is paramount to propose a method to alleviate or solve the problems mentioned above, so as to make a more accurate prediction. Therefore, in this article, we contribute with a neural network that emerges from time-variant framework dedicated to learning complex dynamic dependencies of time series more accurately, with three Self-Attention blocks that enhance the local dynamics learning ability, named as Self-attention-based Time-Variant Neural Networks (SATVNN).

3 Self-attention-based time-variant neural networks

3.1 Problem statement

In this section, we give the formulation of the problem. Time series forecasting usually uses a series of historical values as input. We use β to denote the seasonal period of a given time series. A sliding window of length 3β is utilized to extract samples. The input sequence X can be denoted as $X = x_{t-2\beta+1}, x_{t-2\beta}, \ldots, x_t$, which includes the first 2β data points of the window. In the scope of this work, the last β data points constitute the target values, which can be denoted as $Y = y_t, y_{t+1}, \ldots, y_{t+\beta-1}$. Typically, the problem of time series forecasting can be defined as learning a nonlinear mapping function from the input sequence X to the predicted values $\hat{Y}$, as follows:

$$\hat{Y} = f(X) \tag{1}$$

where $f(\cdot)$ is a nonlinear mapping function.

3.2 Overall framework

To solve above multi-step time series prediction problem, we propose a framework called Self-Attention-based Time-Variant Neural Networks (SATVNN). Figure 1 depicts its overall architecture whose core idea can be summarized as follows: (1) Construct a novel time-variant structure to better learn the dynamic changes of time series on different

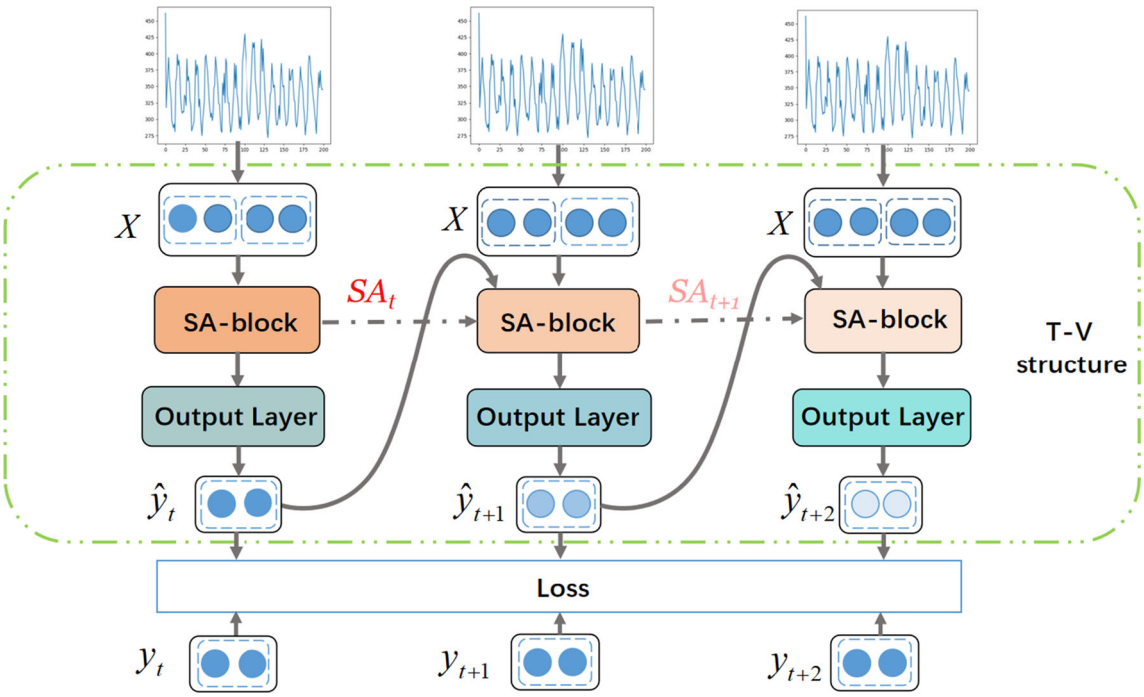

Fig. 1 SATVNN framework. Our SATVNN framework is a novel time-variant (T-V) framework. The heterogeneity over a sequence is represented by Self-Attention blocks (SA-blocks) and outputs of different colors

scales and mitigate vanishing gradients; (2) Design a Self-Attention block to model local temporal dependencies of time series. In the Self-Attention block, Gaussian self-attention, Laplace self-attention and a novel Cauchy self-attention are, respectively, designed to make the proposed framework more flexible to reflect recent changes in temporal data.

3.3 Time-variant structure

Inspired by [15], we design a novel time-variant structure (T-V Structure) to better learn various dynamic scales across multiple time steps and mitigate vanishing gradients. As illustrated in Fig. 1, its overall architecture is composed of three parts, namely a series of inputs, a series of sequentially connected hidden blocks called Self-Attention blocks and a series of outputs. Its network structure between interleaved outputs varies over time in architecture and parameters. The result is that T-V Structure is time variance. Furthermore, the special interleaved output in T-V Structure enables the structure to mitigate vanishing gradient problem. The details of the building parts, Algorithm 1 that shows the procedure of our SATVNN and the time-variant analysis are described in the following paragraphs.

3.3.1 Inputs

Given a set of univariate time series $X = x_{t-2\beta+1}, \ldots, x_t$ as inputs, we feed them into each Self-Attention block in the middle part of the framework. For each input $X \in \mathbb{R}^{B\times L}$, we transform it into $X' \in \mathbb{R}^{B\times L\times d_{in}}$, where B is the batch-size, L denotes the length of input data and here $L = 2\beta$ because the primary goal of our task is to predict the last β data based on the former 2β data. To provide a richer vector representation of the input data, we map the dimension of L from 2β to a higher dimensional space $\mathbb{R}^H$ by a linear layer, which is similar to [35]. We take $\mathbb{R}^H$ as one of the many hyperparameters and select the best hyperparameter $\mathbb{R}^H$ through Section 4.3. We define the last dimension of input data as d_{in}, here $d_{in} = 1$ because the data is univariate. To sum up, the dimension of input data can be expressed as $X' \in \mathbb{R}^{B\times H\times d_{in}}$.

3.3.2 Self-attention block

We build a Self-Attention block (SA-block) to reflect recent changes in temporal data. As one of the most important components of the T-V Structure, each SA-block has its own unique parameters, even if each SA-block's architecture is the same. This is mainly because the T-V Structure is time variance. Figure 2 shows a graphical illustration of SA-block. It mainly consists of an input layer, a positional encoding operation, N_x identical layers

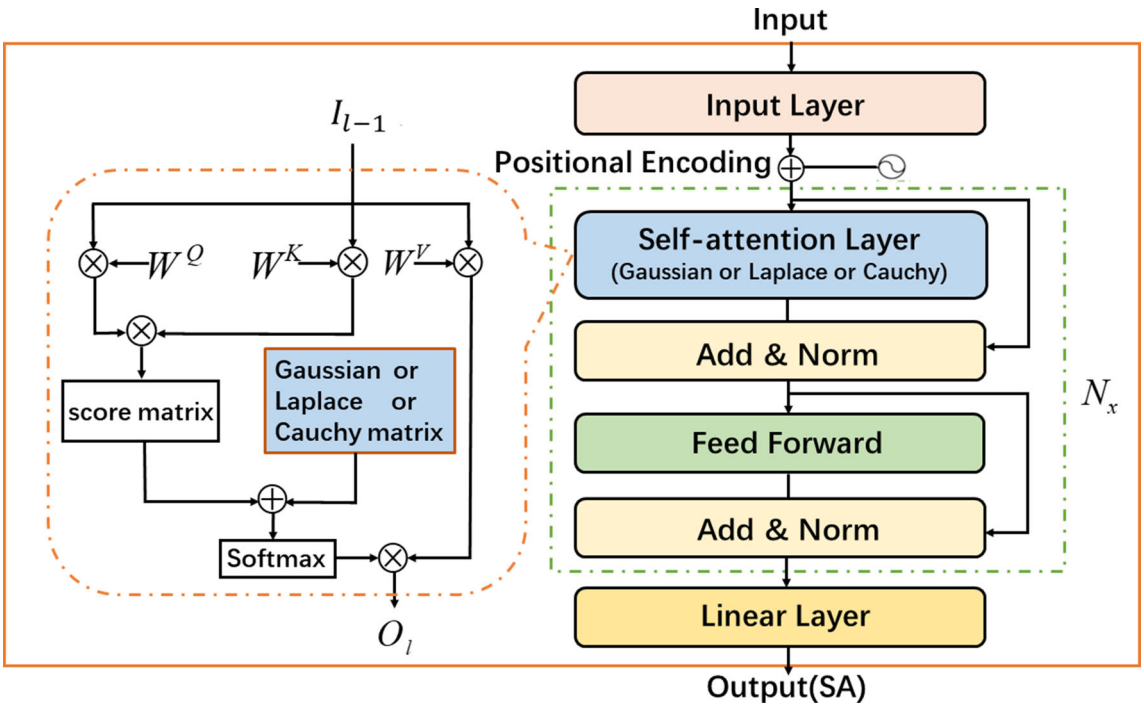

Fig. 2 SA-block. Gaussian, laplace and proposed Cauchy self-attention block can reflect recent changes in temporal data. They are mainly composed of Gaussian, Laplace and Cauchy self-attention layers, respectively

and a linear layer. It is worth noting that each identical layer is mainly composed of two sub-layers: a proposed self-attention layer and a feed-forward layer. The sub-layers are stacked by residual connections and normalization layers as shown in Add and Norm in Fig. 2.

The input layer aims to map the dimension of input vector $X' \in \mathbb{R}^{B\times H\times d_{in}}$ to a d-dimension vector through a fully connected network, which is crucial for SA-block to employ self-attention mechanism. Because proposed SA-block architecture does not contain recurrence, the positional encoding is utilized to encode sequential information by adding a positional encoding vector to an input vector.

The obtained vector $X_{\text{input}} \in \mathbb{R}^{B\times H\times d}$ is then fed into a stack of N_x identical layers. As a core component of each identical layer, the self-attention layer is designed to overcome the shortcoming of the original self-attention that cannot capture the local temporal dependencies. Its architecture is illustrated in the left half of Fig. 2. To enable the framework to better capture recent changes in temporal data, we, respectively, present three self-attention mechanisms, i.e., Gaussian self-attention, Laplace self-attention and our proposed Cauchy self-attention. Their core components are Gaussian distribution, Laplace distribution and Cauchy distribution. Compared with other two distributions, the newly proposed Cauchy distribution is more suitable for reflecting recent changes in time series.

The reason is summarized as follows: given the same data, Cauchy distribution can not only broaden the scope of attention but also avoid the situation that the Gaussian distribution rapidly approaches zero as the distance increases. As the distance increases, the distribution value approaches zero, which means that data with a slightly longer distance are not important, which affects the accuracy of prediction to a certain extent, because there are some distant data related to the current data in time series. Furthermore, compared to Laplace distribution, Cauchy distribution is a more concentrated distribution, which pays more attention to the more recent data. A more detailed account of calculation process of the original self-attention mechanism and three variants of self-attention mechanism is given as follows.

Original self-attention It is now well established from a variety of studies that self-attention plays a critical role in time series prediction. The main calculation flow of the self-attention can be summarized as mapping a query and a group of key-value pairs to the output. Formally, given a input sequence $X_{input} \in \mathbb{R}^{B\times H\times d}$, the self-attention constructs the l-th layer's hidden states by focusing on the hidden states in the $(l-1)$-th layer.[1] The $(l-1)$-th layer is converted into three independent weight matrices, i.e., query $Q_l \in \mathbb{R}^{B\times H\times d}$, key $K_l \in \mathbb{R}^{B\times H\times d}$ and value $V_l \in \mathbb{R}^{B\times H\times d}$ as follows:

[1] The first layer is an input layer.

$$Q_l = W^Q I_{l-1}, \tag{2}$$

$$K_l = W^K I_{l-1}, \tag{3}$$

$$V_l = W^V I_{l-1}, \tag{4}$$

where $I_{l-1} \in \mathbb{R}^{B\times H\times d}$ denotes the $(l-1)$-th hidden layer, $W^Q \in \mathbb{R}^{d\times d}$, $W^K \in \mathbb{R}^{d\times d}$ and $W^V \in \mathbb{R}^{d\times d}$ are trainable parameters, and their initial values are initialized randomly.

The result $S_l(Q_l, K_l) \in \mathbb{R}^{B\times H\times H}$ of the inner product of query and key is considered to be the weights of each location:

$$\mathrm{S}_l(Q_l, K_l) = \frac{Q_l(K_l)^T}{\sqrt{d_k}}, \tag{5}$$

where $\sqrt{d_k}$ denotes the scaling factor. The aim of $\sqrt{d_k}$ is to prevent the result of inner product from being too large, entering the saturation domain of the softmax function and causing the vanishing gradient.

The output of original scaled dot product self-attention $O_l(Q_l, K_l, V_l) \in \mathbb{R}^{B\times H\times d}$ is calculated as a weighted sum of values as follows:

$$O_l(Q_l, K_l, V_l) = \mathrm{softmax}(S_l(Q_l, K_l) \bullet D)V_l. \tag{6}$$

Noted that the softmax operation is utilized to normalize the results to a probability distribution and matrix $D \in \mathbb{R}^{H\times H}$ in which all upper triangular elements are set to $-\infty$ is used to avoid future information leakage.

From the above explanation, we know that the calculation of attention is mainly divided into two parts. One is the calculation of the attention matrix. The attention matrix is obtained by multiplying K and Q (each row of the attention matrix represents the attention of each token relative to other tokens in the sequence, and softmax is added only to normalize the attention). K and Q are calculated using different W^Q and W^K, which can be understood as projections in different spaces. Due to the projection of different spaces, the generalization ability of the calculated attention matrix is higher. The other part is to map the original V to a new space using the attention matrix. V represents the original sequence, so we multiply the attention matrix with V to get a weighted result. In other words, the tokens in V are not related to each other, but V multiplies the attention matrix makes each token in V focus on the part of the interest.

Gaussian self-attention To make original self-attention mechanism have the ability to pay more attention to the scope of the local region, [13] proposed a Gaussian self-attention model for natural language inference. Inspired by this work, we apply Gaussian self-attention to time series prediction for the first time to reflect recent changes in temporal data. As shown in the left half of Fig. 2, Gaussian self-attention is formed by introducing a Gaussian weighting matrix $G(dis) \in \mathbb{R}^{H\times H}$ into the original score matrix. Relationship weights between data in the obtained matrix obey the Gaussian distribution. The probability density of Gaussian distribution is $g(\mathrm{dis}) = e^{-\lambda dis^2}$, where λ is a parameter that the position importance decreases with distance. The position distance between two data is dis. Specifically, $dis = |i-j|$ denotes the distance between the ith data and the jth data. Formally, Gaussian self-attention is as follows:

$$S_l - G(\mathrm{dis}) = G(\mathrm{dis}) + \frac{Q_l(K_l)^T}{\sqrt{d_k}}, \tag{7}$$

$$O_l - G(\mathrm{dis}) = \mathrm{softmax}(S_l - G(\mathrm{dis}) \bullet D)V_l. \tag{8}$$

Laplace self-attention Although Gaussian self-attention effectively improves the importance of neighboring data, the shortcoming of the method is that the importance of data tends to zero rapidly as the distance increases. Another important alternative is Laplace self-attention. Similar to Gaussian self-attention, Laplace weighting matrix can be denoted as $L(\mathrm{dis}) \in \mathbb{R}^{H\times H}$, and relationship weights between data in the matrix obey $l(\mathrm{dis}) = e^{-\lambda dis}$. For Laplace distribution in Laplace self-attention, the importance of data decreases slowly as the distance increases, which retains the importance of non-neighboring data in sequence representation to a some extent, but fails to reflect the fact that adjacent data are of prominent importance. The Laplace self-attention is denoted as follows:

$$S_l - L(\mathrm{dis}) = L(\mathrm{dis}) + \frac{Q_l(K_l)^T}{\sqrt{d_k}}, \tag{9}$$

$$O_l - L(\mathrm{dis}) = \mathrm{softmax}(S_l - L(\mathrm{dis}) \bullet D)V_l. \tag{10}$$

Cauchy self-attention Consequently, we propose a Cauchy self-attention, which can effectively improve the importance of adjacent data, and the importance of data will also slowly decrease as the distance increases rather than rapidly approaching zero. As shown in the left half of Fig. 2, Cauchy self-attention is formed by introducing a Cauchy weighting matrix $C(\mathrm{dis}) \in \mathbb{R}^{H\times H}$ into original score matrix. Cauchy weighting matrix $C(\mathrm{dis})$ is calculated as follows:

$$C(\mathrm{dis}) = \begin{bmatrix} c_{1,1} & c_{1,2} & \cdots c_{1,H} \\ c_{2,1} & c_{2,2} & \cdots c_{2,H} \\ \vdots & & \\ c_{H,1} & c_{H,2} & \cdots c_{H,H} \end{bmatrix}, \tag{11}$$

where $c_{i,j}$ is $\frac{1}{1+\lambda|i-j|^2}$, which follows Cauchy distribution.

That is, Cauchy matrix is utilized to pay more attention to the local region around the current data, and the attention

weights become smaller as the distance increases. To highlight the advantages of Cauchy distribution more intuitively, graphical illustration of three distributions is given in Fig. 3. The Gaussian distribution curve shows that the importance of the data decreases rapidly. The importance of data approaches zero when the distance is greater than 2, which suppresses the representation of the current data by the slightly distant data. For Laplace distribution, the orange curve decays down too slowly to reflect the prominent importance of adjacent data well. In contrast, Cauchy distribution, as shown in the green curve, falls more rapidly than orange curve, rather than approaching zero as quickly as blue curve. This makes the model not only pay more attention to adjacent data, but also pay less attention to remote data. The proposed Cauchy self-attention is denoted as follows:

$$S_l - C(\text{dis}) = C(\text{dis}) + \frac{Q_l(K_l)^T}{\sqrt{d_k}}, \tag{12}$$

$$O_l - C(\text{dis}) = \text{softmax}(S_l - C(\text{dis}) \bullet D)V_l. \tag{13}$$

We add a feed-forward layer after the output of the self-attention layer to enhance the expressiveness of the model. As shown in Fig. 2, the input of the feed-forward layer is one of the following three expressions:

$$A_l - G = \text{LayerNorm}(O_l - G(\text{dis}) + X_{\text{input}}), \tag{14}$$

$$A_l - L = \text{LayerNorm}(O_l - L(\text{dis}) + X_{\text{input}}), \tag{15}$$

$$A_l - C = \text{LayerNorm}(O_l - C(\text{dis}) + X_{\text{input}}), \tag{16}$$

where LayerNorm denotes the normalization operation in Add and Norm. Take $A_l - C \in \mathbb{R}^{B\times H\times d}$ as an example, the feed-forward layer that consists of two fully connected layers and ReLU activation function can be formulated as follows:

$$F_l - C = \max(0, (A_l - C)W_1 + b_1)W_2 + b_2, \tag{17}$$

where $F_l - C \in \mathbb{R}^{B\times H\times d}$ denotes the transformed space of the self-attention layer's output, $W_1 \in \mathbb{R}^{d\times d_{ff}}$, $b_1 \in \mathbb{R}^{d_{ff}}$, $W_2 \in \mathbb{R}^{d_{ff}\times d}$, $b_2 \in \mathbb{R}^{d}$, where d is the dimension of the input vector transformed by the input layer and d_{ff} is the dimension of the middle layer of the two fully connected layers. In our model, $d_{ff} = 3d$.

Then, we employ an output layer shown in Fig. 1 to generate the final prediction. Formally,

$$\hat{y}_t = (\widehat{F}_l - C)W_3 + b_3. \tag{18}$$

where $\hat{y}_t$ is the prediction at time t. $\widehat{F}_l - C$ denotes the transformed value after the Add and Norm operation on $F_l - C \in \mathbb{R}^{B\times H\times d}$. Note that the dimension of $\widehat{F}_l - C$ is originally the same as that of $F_l - C$. But here $\widehat{F}_l - C \in \mathbb{R}^{B\times Hd_{\text{out}}}$, because we transform $\widehat{F}_l - C \in \mathbb{R}^{B\times H\times d}$ into $\widehat{F}_l - C' \in \mathbb{R}^{B\times H\times d_{\text{out}}}$ through a linear layer shown in Fig. 2 first, and then, we reshape $\widehat{F}_l - C' \in \mathbb{R}^{B\times H\times d_{\text{out}}}$ into $\widehat{F}_l - C'_{\text{reshape}} \in \mathbb{R}^{B\times Hd_{\text{out}}}$, where d_{out} is the output dimension. $W_3 \in \mathbb{R}^{Hd_{\text{out}}\times d_{\text{out}}}$, $b_3 \in \mathbb{R}^{d_{\text{out}}}$,

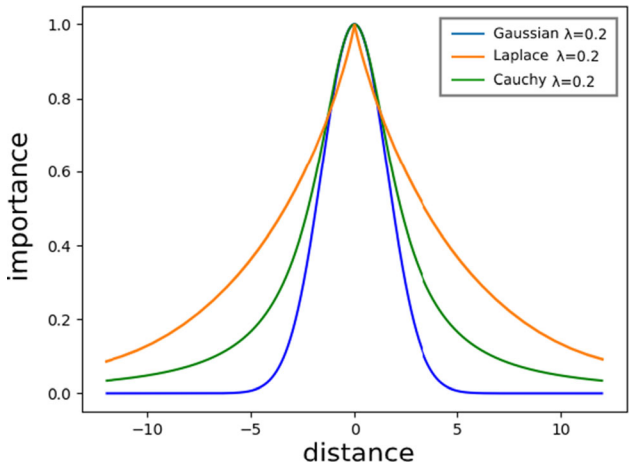

Fig. 3 Decreasing trend of the importance of data with the increase in distance

3.3.3 Output

Each output of each self-attention block is a predicted value for each future step. Of particular concern is that our goal is to provide sequential level forecasts. To achieve this goal, the most intuitive idea is to connect multiple SA-blocks. The final sequential level predicted values can be denoted as:

$$\hat{Y} = \left[\hat{y}_t, \hat{y}_{t+1}, \ldots, \hat{y}_{t+\beta-1}\right]. \tag{19}$$

Noted that the more output, the deeper the network, the easier it is for the gradients to disappear/explode. Vanishing gradient problem is mainly caused by the repeated application of chain rule when calculating the gradients. A chain of factors are produced by the chain rule. SATVNN mitigates this problem because it breaks the chain into the sum of a series of factors by interleaving the outputs between SA-blocks, which is more stable than performing a product operation on a series of factors. That is to say, the sum of the smaller factors is less likely to approach zero (vanishing) than their product. This makes the prediction $\hat{Y}$ more accurate.

Algorithm 1: SATVNN Algorithm

Input: A set of univariate time series : $X = x_{t-2\beta+1}, x_{t-2\beta}, \ldots, x_t$; the seasonal period of given time series : β, the parameter of model: θ.

for *each training iteration* **do**

  for m *steps* **do**

    Construct training samples

    Compute $\hat{y}_t$ by the first SA-block SA_1 and Output layer L_1:

    $\hat{y}_t = L_1(SA_1(X))$

    Concatenate input X, the predicted value of the previous step $\hat{y}_t$ and hidden state in the previous step SA_t, then put them $(X \circ \hat{y}_t \circ SA_t)$ into second SA-block SA_2, Output layer L_2:

    $\hat{y}_{t+1} = L_2(SA_2\,(X \circ \hat{y}_t \circ SA_t))$

    Compute $\hat{y}_{t+n}$ as described above:

    $\hat{y}_{t+n} = L_{n+1}\,(SA_{n+1}\,(X \circ \hat{y}_{t+n-1} \circ SA_{t+n-1}))$

  $L_\Theta = \|y_{t+n} - \hat{y}_{t+n}\|_2^2$

  Optimize the θ over total loss L_Θ.

3.3.4 Time-variant analysis

SATVNN is a novel time-variant structure, which can better learn the dynamic changes of time series on different scales, especially in capturing the dynamic changes of the recent data. The time variance of SATVNN is embodied in the change of network structure between interleaved outputs with time. The concrete proof of the time variance is given below.

As illustrated in Fig. 4, SATVNN with input X_t, hidden state SA_t and predicted value $\hat{y}_t$ at time t given by

$$\mathrm{SA}_t = f_t(X_t, \mathrm{SA}_{t-1}, \hat{y}_{t-1}), \tag{20}$$

$$\hat{y}_t = g_t(\mathrm{SA}_t) \tag{21}$$

is time variance, where f_t represents a SA-block at time t, g_t is the Output layer at time t, SA_{t-1} is the hidden state at time $t-1$ and $\hat{y}_{t-1}$ is the predicted values at time $t-1$.

According to Eqs. (20) and (21), $\hat{y}_t$ can be represented as:

$$\hat{y}_t = g_t(f_t(X_t, \mathrm{SA}_{t-1}, \hat{y}_{t-1})). \tag{22}$$

Proof Given input X_t, hidden state SA_t and predicted value $\hat{y}_t$, X_t will be replaced by the following expression $X'_t = X_{t-t_0}$ when time is shifted by t_0. Similarly, let $\mathrm{SA}'_t = \mathrm{SA}_{t-t_0}$. Time invariance requires $\hat{y}_{t-t_0} = \hat{y}_t'$. The reason is that a time-invariant system is defined as a system in which the time shift of the input sequence causes the corresponding shift in the output sequence [48]. Then, according to Eq. (22), $\hat{y}_{t-t_0}$ is given by:

$$\hat{y}_{t-t_0} = g_{t-t_0}\left(f_{t-t_0}\left(X_{t-t_0}, SA_{t-t_0-1}, \hat{y}_{t-t_0-1}\right)\right), \tag{23}$$

and $\hat{y}_t'$ is formed by

$$\begin{aligned}\hat{y}_t' &= g'_t\left(f'_t\left(X'_t, \mathrm{SA}'_{t-1}, \hat{y}'_{t-1}\right)\right)\\ &= g'_t\left(f'_t\left(X_{t-t_0}, \mathrm{SA}_{t-t_0-1}, \hat{y}'_{t-1}\right)\right).\end{aligned} \tag{24}$$

where $\hat{y}'_t$ is the predicted value corresponding to the input X'_t and $\hat{y}'_{t-1}$ is the predicted value of the previous step.

The above derivation process proves that the SATVNN is time variance, because $\hat{y}_{t-t_0} \neq \hat{y}_t'$. □

SATVNN is time variance because its parameters and architecture vary over time on input and output sequences. Comparatively, RNN has fixed parameters that are repeated at each time step, resulting in a time-invariant model. In order to better understand the time variance of SATVNN, we go further into the time invariance proof of RNN as follows.

RNN with input X_t, hidden state h_t and predicted value $\hat{y}_t$ at time t given by

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

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

is time invariance, where f is a hidden layer, g is the output

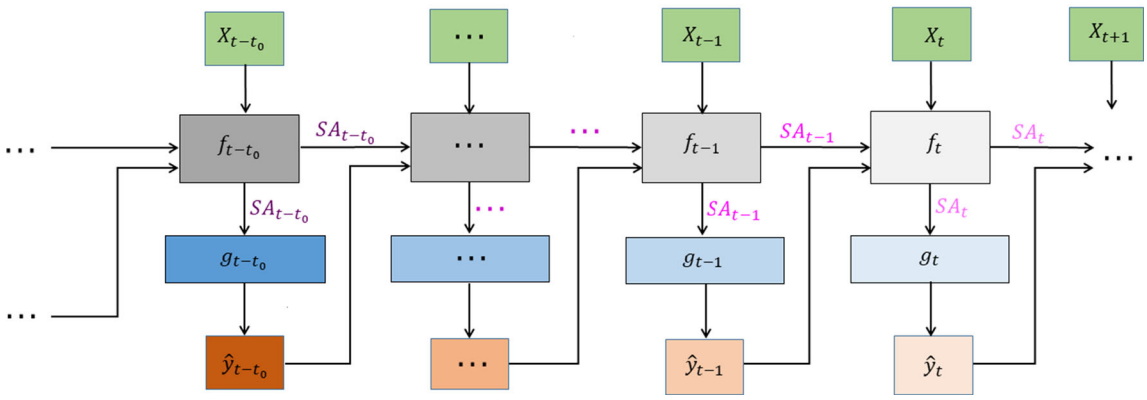

Fig. 4 Graphical illustration of time-variant property of SATVNN. The network structure between interleaved outputs varies over time in architecture and parameters. We utilize different colored SA-blocks (f_t), different colored output layers (g_t) and different colored outputs ($\hat{y}_t$) to denote the heterogeneity over a sequence

layer and h_{t-1} denotes the hidden state at time $t-1$. According to Eqs. (25) and (26), we obtain $\hat{y_t}$ as follows:

$$\hat{y_t} = g(f(X_t, h_{t-1})). \tag{27}$$

□

Proof Given input X_t, hidden state h_t and predicted value $\hat{y_t}$, X_t will be replaced by the following expression $X'_t = X_{t-t_0}$ when time is shifted by t_0. Similarly, let $h'_t = h_{t-t_0}$. Time invariance requires $\hat{y}_{t-t_0} = \hat{y_t}'$. □

Then, according to Eq. (27), $\hat{y}_{t-t_0}$ is given as follows:

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

and $\hat{y_t}'$ is given by:

$$\begin{aligned} \hat{y_t}' &= g\big(f\big(X'_t, h'_{t-1},\big)\big) \\ &= g(f(X_{t-t_0}, h_{t-t_0-1})). \end{aligned} \tag{29}$$

Thus, RNN is time invariant because $\hat{y}_{t-t_0}$=$\hat{y_t}'$. □

3.4 Loss function

Our proposed SATVNN framework needs to be trained by minimizing the mean squared error (MSE) loss function, which is used by a variety of traditional forecasting methods to evaluate the difference between the true values and the predicted values. The optimization objective can be denoted as follows:

$$L_\Theta = \left\| Y - \hat{Y} \right\|_2^2, \tag{30}$$

where θ denotes the learning parameters.

4 Experiments

In this section, we first describe the real-world datasets. Then a set of baseline models for comparison and the experimental settings are provided. To demonstrate the effectiveness of Cauchy self-attention, we compare the performance of Gaussian self-attention-based SATVNN (SATVNN-G), Laplace self-attention-based SATVNN (SATVNN-L) and Cauchy self-attention-based SATVNN (SATVNN-C) on seven real-world datasets. Additionally, the model with the best performance is selected for further comparison with other baseline models. At the end of this section, we conduct some case studies and display the forecast results graphically.

4.1 Data description

Seven real-world datasets are collected from [49] to evaluate our models. We note that there are significant factors captured in any of these datasets, such as periodicity, varying seasonal shape, varying seasonal amplitude and noise. Additional detailed statistics of these datasets are shown in Table 1. In our experiments, we normalize the datasets used for training and test, in which the last 10% of the datasets are used for test.

4.2 Baseline methods

For fairness, each model uses the same input and Mean Square Error (MSE) loss function. A range of baseline models are as follows:

- *SARIMA* [50]: Seasonal Autoregressive Integrated Moving Average is one of the most representative methods in traditional time series forecasting field. It is suitable for forecasting time series with tendency and periodicity.
- *DLM* [51]: Dynamic Linear Model can recursively estimate and predict through Kalman filter.
- *Seq2Seq* [24]: Seq2Seq is a neural network of Encoder–Decoder structure. The encoder can encode sequence information of any length into a vector c. After obtaining the context information vector c, the decoder can decode the information and output it as a sequence.
- *Seq2seq-att* [52]: Attention is used for improving the effect of RNN based encoder–decoder model. Attention gives the model the ability to discriminate, which makes neural network models more flexible to be learned.
- *DeepAR* [20]: It is a model based on Autoregressive Recurrent Neural Network. It can effectively learn the global dependencies from the relevant time series and can learn complex patterns by training the autoregressive recursive networks.
- *TCN* [53]: Temporal Convolutional Network is a variant of convolution neural network. It utilizes dilated convolutions to obtain the global information of the whole sequence and expand the receptive field.
- *ForecastNet* [15]: A deep feed-forward architecture for multi-step time-series forecasting. The structure and parameters of ForecastNet change over time, which is different from traditional networks such as RNN and CNN. Of the four variants of ForecastNet, the one named cFN2 with a linear output layer performs best.

4.3 Parameter settings and sensitivity

The hyperparameters of our three proposed models include the number of encoding blocks N_x, batch size B, epoch τ and adjustable parameter λ. Considering both the performance and efficiency, we set $N_x = 2$, $B = 16$, $\tau = 100$, and $\lambda = \frac{1}{3}$. In addition, we take three proposed models with a

Table 1 Dataset properties

Dataset	Max	Min	Mean	SD	Resolution	Length
Niagra	7610.0	1860.0	5574.54	1186.47	Monthly	1834
Ozone	462.0	266.0	338.02	38.37	Monthly	482
Pphil	401.8	2.3	89.79	47.85	Monthly	1572
River-flow	66,500	3290	23,310.24	13179.27	Monthly	1368
Ssack	1919.88	28.23	277.38	277.09	Monthly	780
Weather (England temperature)	18.8	− 3.1	9.22	4.79	Monthly	2976
Water-usage	226.3	76.83	118.76	26.31	Monthly	276

richer vector dimension of the length of input data $\mathbb{R}^H = 75$ and the last dimension of input data $\mathbb{R}^d = 70$ which confirmed by a grid search. Furthermore, diagrams drawn in Figs. 5, 6 and 7 partially display the parameter sensitivity of the three proposed models in seven datasets.

4.4 Experimental comparison of different self-attention mechanisms

Table 2 summarizes the prediction results obtained by the time-variant neural networks with three Self-Attention mechanisms on the seven datasets. We use the Mean Absolute Scaled Error (MASE) [54] and the Symmetric Mean Absolute Percentage Error (SMAPE) [55] to evaluate the performance of these models. The results indicate that SATVNN-C outperforms the other two models on six real-world datasets except for the England temperature dataset. Even for the England temperature dataset, only the SMAPE of SATVNN-C is slightly higher than that of the SAVTNN-L model, but the MASE of SATVNN-C is still lower than the other two models. Furthermore, some representative examples are given to verify the effectiveness of SATVNN-C. Specifically, as illustrated in Fig. 8, the MASE of SATVNN-C on the England temperature dataset is 2.8% and 2.4% lower than that of SATVNN-G and SATVNN-L, respectively. For Ozone dataset, the MASE of SATVN-N-C is 3.5% and 5.0% lower than that of SATVNN-G and SATVNN-L as visualized in Fig. 9. What's more, the results shown in Fig. 10 demonstrate that SMAPE of SATVNN-C is 9.9% and 6.8% lower than that of SATVNN-G and SATVNN-L on the Niagra dataset. For the remaining datasets, SATVNN-C also achieves results from 0.1 to 4% lower than the other two models.

That is because Gaussian distribution in SATVNN-G shields the contribution of non-adjacent data to the current data, while Laplace distribution in SATVNN-L does not adequately emphasize the prominent importance of adjacent data. For Cauchy distribution in SATVNN-C, it not only retains the relative importance of non-adjacent data to the current data but also further highlights the importance of adjacent data.

In addition to comparing the forecasting accuracy, we also conduct further experimental investigations to estimate the running time of the three models on seven datasets. Table 3 shows that less running time is taken by our proposed SATVNN-C per epoch than the other two

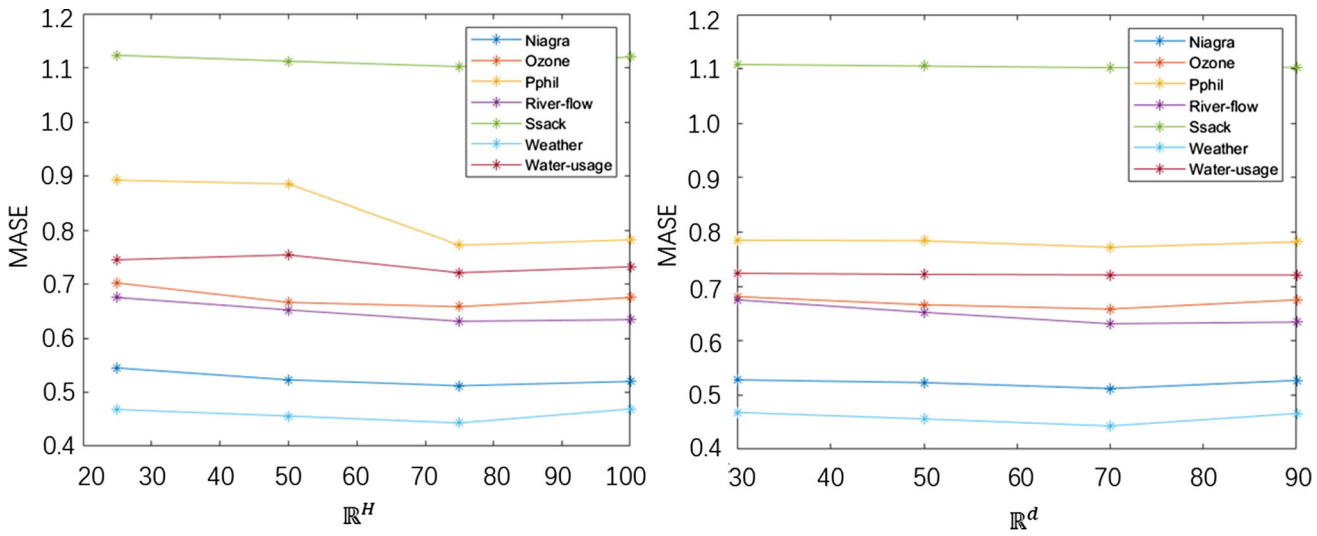

Fig. 5 Parameter sensitivity of SATVNN-C model in seven datasets

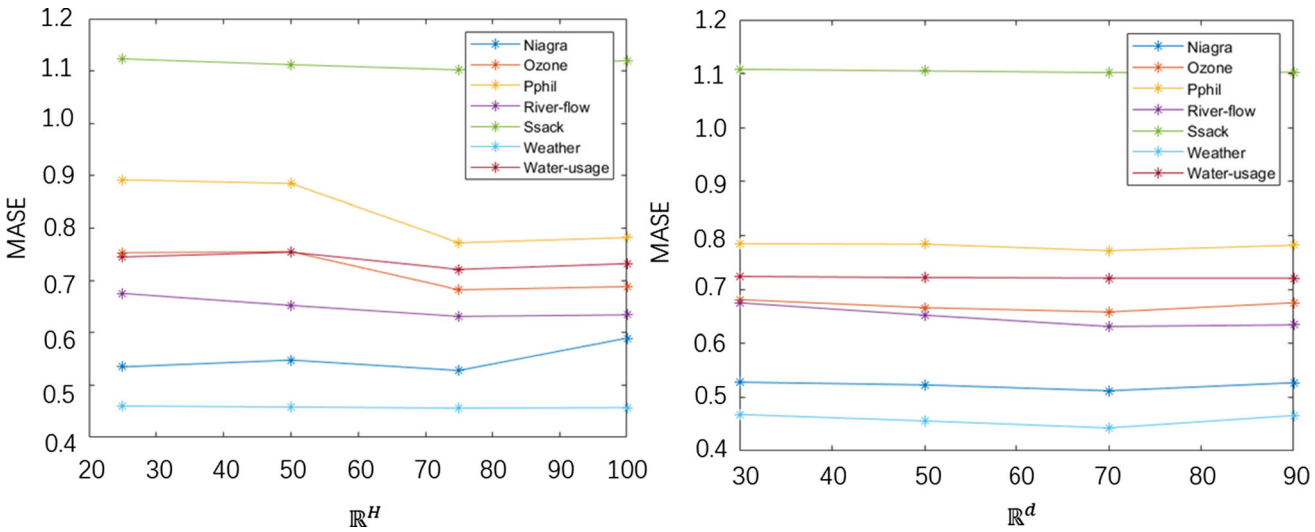

Fig. 6 Parameter sensitivity of SATVNN-G model in seven datasets

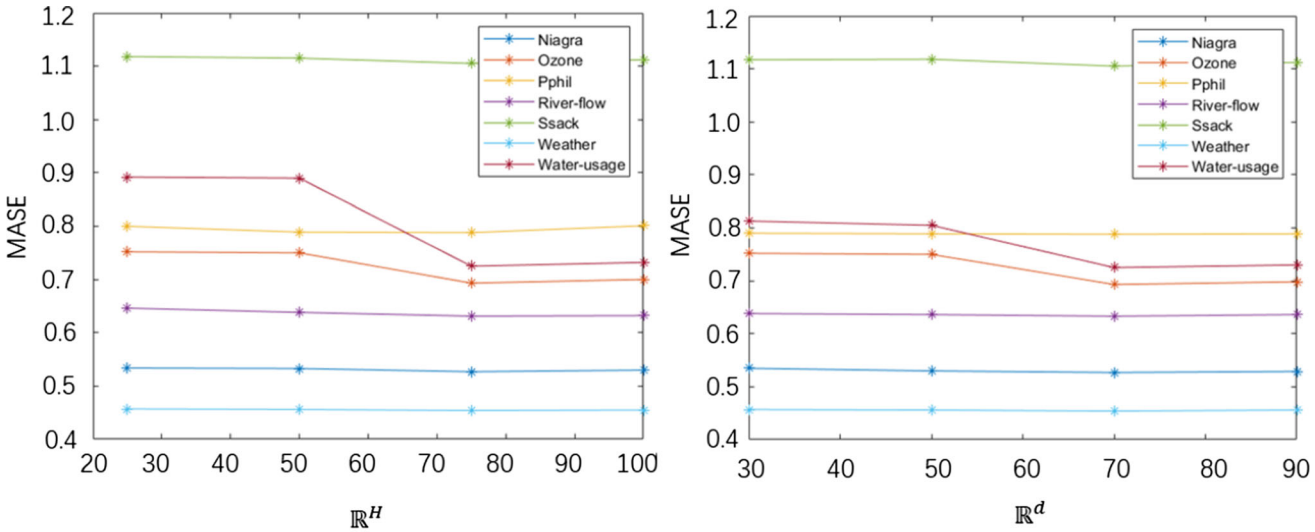

Fig. 7 Parameter sensitivity of SATVNN-L model in seven datasets

Table 2 Results of different self-attention mechanisms based on time-variant neural networks

Dataset		Niagra	Ozone	Pphil	River-flow	Ssack	Weather (England temperature)	Water-usage
Methods	Metrics							
SATVNN-C	MASE	**0.512**	**0.658**	**0.772**	**0.631**	**1.102**	**0.443**	**0.721**
	SMAPE	**11.452**	**27.884**	**42.231**	**26.622**	**54.418**	10.784	**7.142**
SATVNN-G	MASE	0.528	0.682	0.785	0.634	1.105	0.456	0.752
	SMAPE	12.712	27.953	42.525	26.954	54.486	10.825	7.183
SATVNN-L	MASE	0.527	0.693	0.788	0.633	1.115	0.454	0.725
	SMAPE	12.295	27.967	42.286	26.988	56.214	**10.741**	7.325

approaches. The main reason for this is that Gaussian and Laplace distributions have the exponential computational complexity of e, while our distribution does not have such computational complexity. Taken together, SATVNN can

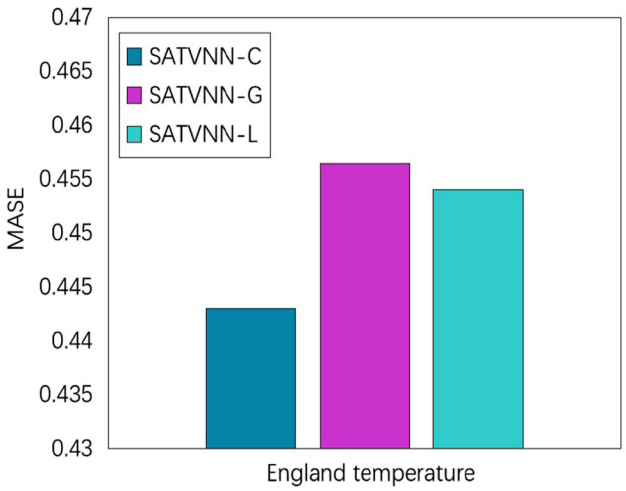

Fig. 8 MASE of three versions of our proposed model on the England temperature dataset

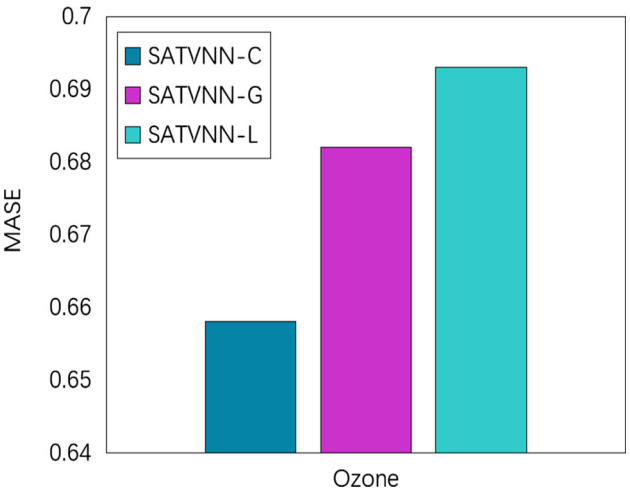

Fig. 9 MASE of three versions of our proposed model on the Ozone dataset

provide more reliable input through SA-block, and our proposed Cauchy self-attention block enables SATVNN to produce more accurate prediction results faster.

4.5 Experimental results of SATVNN-C compared with baseline methods

Table 4 records the experimental results of the proposed SATVNN-C model and the baseline models running 100 epochs on seven datasets. It can be seen that although SARIMA and DLM models are both linear models [56], they can trivially obtain a high performance which is very close to the best result on the Pphil data. The main reason for this result is that the dynamics of the Pphil data are more linear. However, DLM and SARIMA perform poorly

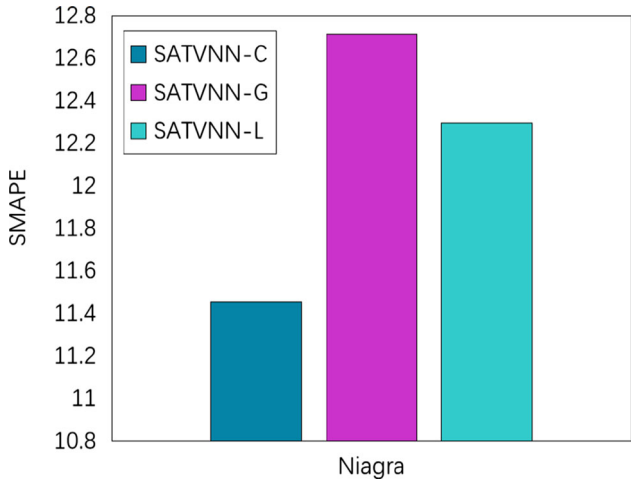

Fig. 10 SMAPE of three versions of our proposed model on the Niagra dataset

on such datasets with periodic shapes and amplitudes changing over time.

We obtain another important insight from Table 4 that DeepAR performs poorly on several datasets. This is mainly because this model cannot well capture the dynamic changes of these datasets with different amplitudes, different seasonal shapes and complex trends, such as the Water-Usage dataset and the Ssack dataset. TCN achieves low accuracy on most datasets. The reason for its poor performance may be performing dilated convolutions with fewer samples.

As one of the commonly used time series forecasting models in deep learning, Seq2Seq-att model shows superior performance on the Ozone dataset due to its unique mechanism that enables the model to capture long-term dependencies of time series. On the remaining datasets, the proposed model achieves competitive performance. This reveals that in addition to capturing the long-term dependencies of time series, it is necessary to further learn the local dependencies of temporal data.

As a powerful competitive model of our proposed SATVNN-C model, ForecastNet provides better results than other linear and nonlinear models, which is mainly due to its unique time-variant structure. However, it performs worse on several datasets than our proposed SATVNN-C model, which proves that the proposed time-variant SATVNN-C model can more flexibly capture the dynamic changes of complex nonlinear temporal data on different scales than ForecastNet and also demonstrates that the newly proposed time-variant structure makes a greater contribution to improving the prediction performance than original time-variant structure.

Table 3 Running time of SATVNN-G, SATVNN-L and SATVNN-C on seven datasets (1 epoch)

Datasets	River-flow	Ozone	Niagra	Weather (England temperature)	Water-usage	Pphil	Ssack
SATVNN-G	9.193	6.534	11.923	18.558	2.544	18.322	9.504
SATVNN-L	9.002	5.699	11.682	18.443	2.455	18.303	9.025
SATVNN-C	**8.811**	**5.650**	**11.678**	**18.301**	**2.417**	**17.982**	**8.823**

Table 4 Results of SATVNN-C and other baseline models

Dataset	Metrics	SARIMA	DLM	Seq2Seq	Seq2Seq-attn	TCN	DeepAR	ForecstNet (cFN2)	SATVNN-C
Niagra	MASE	0.891	0.735	0.542	0.591	0.558	0.538	0.535	**0.512**
	SMAPE	22.562	30.267	12.364	13.983	15.852	15.624	12.268	**11.453**
Ozone	MASE	0.872	0.787	1.534	**0.575**	0.687	1.013	0.755	0.652
	SMAPE	36.534	31.212	59.721	28.018	33.853	41.814	36.035	**27.883**
Pphil	MASE	0.792	0.834	0.893	1.102	0.951	1.025	0.788	**0.772**
	SMAPE	42.253	45.621	49.842	47.523	45.828	46.121	42.268	**42.231**
River-flow	MASE	1.152	1.259	0.878	1.801	1.138	1.122	0.636	**0.631**
	SMAPE	32.263	28.265	31.628	29.132	38.693	35.261	26.785	**26.623**
Ssack	MASE	1.787	1.622	1.304	1.223	1.422	1.519	1.119	**1.102**
	SMAPE	69.355	69.951	59.362	62.653	59.891	68.235	58.571	**54.413**
England temperature	MASE	0.644	0.551	0.482	0.408	0.491	0.512	0.457	**0.443**
	SMAPE	25.589	19.632	16.813	19.565	18.967	18.265	10.876	**10.783**
Water-usage	MASE	1.327	1.358	1.363	1.252	1.154	1.225	0.895	**0.722**
	SMAPE	7.362	8.691	9.857	12.524	13.252	17.167	8.515	**7.142**

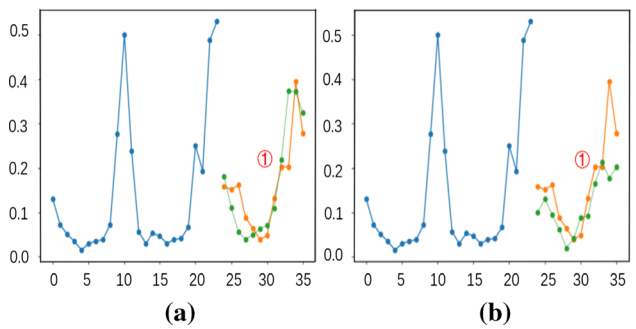

Fig. 11 Results of SATVNN-C (**a**) and ForecastNet (**b**) on the Ssack dataset

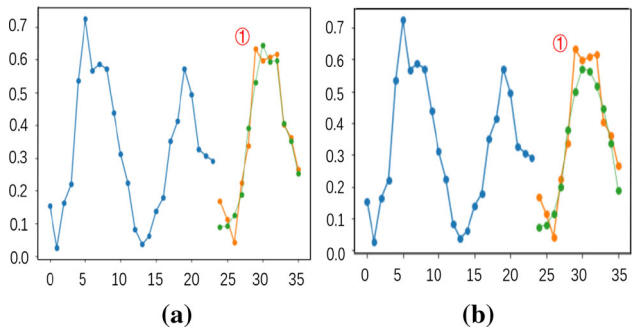

Fig. 12 Results of SATVNN-C (**a**) and ForecastNet (**b**) on the Ozone dataset

4.6 Case study

In this section, we display the prediction results of ForecastNet and SATVNN-C on several datasets in graphical form to illustrate the benefit of our proposed model in capturing the dynamic changes of time series on different scales, especially in capturing the dynamic changes of the recent data. Figures 11, 12 and 13 investigate the forecasting performance of SATVNN-C (part (a)) and ForecastNet (part (b)) on the Ssack, Ozone and Niagra dataset. Figure 11 shows that the first, second and third cycle patterns of the Ssack dataset are different. The blue curve denotes the real values of the first two periods, and the real values and predicted values of the next period are represented by orange curve and green curve, respectively. As depicted in Fig. 11, ForecastNet model is not as

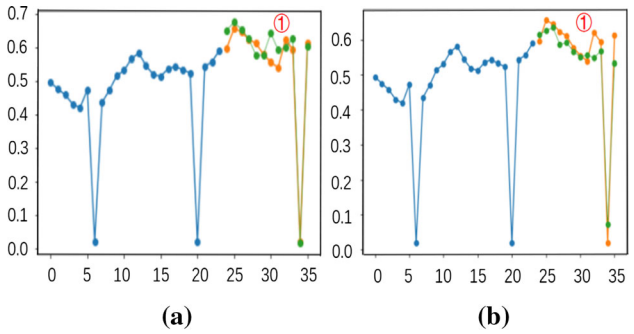

Fig. 13 Results of SATVNN-C (**a**) and ForecastNet (**b**) on the Niagra dataset

sensitive as our proposed model in capturing the latest data changes. Specifically, it is difficult for ForecastNet to reach the peak of the next forecast period, and the predicted values after the marked point ① are quite different from the ground truth. On the contrary, our model can capture the change of the peak value well, and the predicted values after the marked point ① are basically consistent with the real values. Similar situations also occur in Figs. 12 and 13.

Knowing whether a developed method can be successful in predicting only one step, or whether it is successful also in predicting more than one step can provide a good view of the success of the program. Therefore, we also compare the performance of ForecastNet and SATVNN-C for forecasting the next 3 or 6 data when time $t = 0$. We record the results in Tables 5 and 6. With in-depth analysis, these examples demonstrate that the SATVNN-C model has a stronger ability to capture the dynamic changes of time series on different scales than ForecastNet.

Table 5 Results of ForecastNet and SATVNN-C for forecasting next 3 data

Dataset	Metrics	ForecastNet (cFN2)	SATVNN-C
Niagra	MASE	0.511	**0.494**
	SMAPE	12.238	**11.281**
Ozone	MASE	0.650	**0.646**
	SMAPE	26.750	**26.384**
Pphil	MASE	0.777	**0.774**
	SMAPE	42.353	**42.218**
River-flow	MASE	0.639	**0.635**
	SMAPE	**25.858**	26.702
Ssack	MASE	1.123	**1.075**
	SMAPE	59.745	**58.085**
England temperature	MASE	0.452	**0.448**
	SMAPE	11.022	**10.852**
Water-usage	MASE	0.751	**0.745**
	SMAPE	7.253	**7.216**

Table 6 Results of ForecastNet and SATVNN-C for forecasting next 6 data

Dataset	Metrics	ForecastNet (cFN2)	SATVNN-C
Niagra	MASE	0.523	**0.510**
	SMAPE	12.628	**11.627**
Ozone	MASE	0.655	**0.643**
	SMAPE	26.921	**26.111**
Pphil	MASE	0.781	**0.767**
	SMAPE	42.492	**41.990**
River-flow	MASE	0.634	**0.626**
	SMAPE	26.166	**26.006**
Ssack	MASE	1.117	**1.073**
	SMAPE	57.446	**56.508**
England temperature	MASE	0.451	**0.446**
	SMAPE	10.987	**10.841**
Water-usage	MASE	0.731	**0.725**
	SMAPE	7.226	**7.018**

5 Conclusion

This paper implements a novel time-variant framework named SATVNN for multi-step time series and generally capable of capturing dynamic changes of time series on different scales more accurately with its time-variant structure. Three versions are derived from this framework, which have different self-attention blocks whose goals are capturing local dynamics of temporal sequences, to make the model better reflect the recent changes in time series. Within our conducted experiments, the proposed SATVNN-C shows promising results compared with baseline models. In addition, the other two versions with Gaussian and Laplace self-attention block also obtain good forecasting accuracy. For our future work, we intend to explore the use of memory reduction techniques because the proposed models may need more memory by avoiding sharing parameters.

Appendix

Proof of convergence of SATVNN

The weight of the time-invariant network is often constant, and the mapping between the input and output of the network is fixed after training. However, the mapping relationship between the input and output of the time-variant

system will change with time. In this paper, we propose a SATVNN with time-variant weights, which is used to establish a neural network each time, and the neural network at that time is used to approach the mapping relationship between the input and output of the system at that time. We regard time-variant weight learning as an unknown time-variant parameter estimation problem for nonlinear time-variant systems. In [57], an iterative learning least squares algorithm is proposed to train the weights of the time-variant network. Theoretical analysis shows that the iterative learning least squares algorithm can ensure that the modeling error converges to zero along the iteration axis in the whole time interval. We will elaborate on the reasons in the following paragraphs.

Time-variant neural network The time-variant neural network whose input, output and weight change with time can be expressed as follows:

$$y(t) = \mathcal{E}^{\mathrm{T}}(x(t))W(t) \tag{31}$$

or

$$y(t) = \mathcal{E}^{\mathrm{T}}(x(t), t)W(t) \tag{32}$$

where $x(t) = [x_1(t), x_2(t), \ldots, x_n(t)]^{\mathrm{T}} \in \mathbb{R}^n$ is input vector. $\mathcal{E}(x(t)) = [\mathcal{E}_1(x(t)), \mathcal{E}_2(x(t)), \ldots, \mathcal{E}_l(x(t))]^{\mathrm{T}}$ is a set of activation function vectors. $y(t)$ is output. $W(t) = [w_1(t), \ldots, w_l(t)]^{\mathrm{T}}$ is an adjustable time-variant weight vector. l is the number of neuron nodes.

Time-variant system Let the nonlinear discrete time-variant system be:

$$y_m(t) = f(x_m(t), t) \tag{33}$$

where f is a nonlinear time-variant function, $x_m(t)$ is input vector. $t \in \{0, 1, \ldots, N\}$ is a finite time interval. $m \in \{0, 1, \ldots\}$ is the number of iterations and $y_m(t)$ is output. In the following discussion, we all assume that $x_m(t)$ is bounded.

Iterative learning least square method Let the nonlinear time-variant system (33) be run repeatedly for m time from the $0th$ time, and the input and output data pairs $\{(x_i(t), y_i(t)), t = 0, 1, \ldots, N, i = 0, 1, \ldots, m\}$ are generated by the system. We will use the time-variant neural network (32) to approximate the above input–output mapping.

First, we define $E_m(t) = [\mathcal{E}_0^{\mathrm{T}}(t), \mathcal{E}_1^{\mathrm{T}}(t), \ldots, \mathcal{E}_m^{\mathrm{T}}(t)]^{\mathrm{T}}$ and $Y_m(t) = [y_0(t), y_1(t), \ldots, y_m(t)]^{\mathrm{T}}$,where $\mathcal{E}_i(t) = \mathcal{E}(x_i(t), t)$, according to Eq. (32),

$$Y_m(t) = E_m(t)W(t) \tag{34}$$

We hope to find out the least square solution for $W(t)$, $\hat{W}_m(t)$, so that the following loss function is minimized.

$$\begin{aligned} J_m(\hat{W}_m(t), t) = \frac{1}{2}[Y_m(t) - E_m(t)\hat{W}_m(t)]^{\mathrm{T}} \\ [Y_m(t) - E_m(t)\hat{W}_m(t)] \end{aligned} \tag{35}$$

Write Eq. (35) further as:

$$\begin{aligned} &2J_m\left(\hat{W}_m(t), t\right) = Y_m^{\mathrm{T}}(t)\left[I - E_m(t)\left(E_m^{\mathrm{T}}(t)E_m\right)^{-1}(t)E_m^{\mathrm{T}}(t)\right] \\ &Y_m(t) + \left[\hat{W}_m(t) - \left[E_m^{\mathrm{T}}(t)E_m(t)\right]^{-1}E_m^{\mathrm{T}}(t)Y_m(t)\right]^{\mathrm{T}} \\ &E_m^{\mathrm{T}}(t)E_m(t)\left[\hat{W}_m(t) - \left(E_m^{\mathrm{T}}(t)E_m(t)\right)^{-1}E_m^{\mathrm{T}}(t)Y_m(t+1)\right] \end{aligned} \tag{36}$$

Note that only the second term on the right side of the above formula is related to $\hat{W}_m(t)$, set this item zero. We assume that $E_m^{\mathrm{T}}(t)E_m(t)$ is invertible and achieve the minimum value:

$$\hat{W}_m(t) = \left(E_m^{\mathrm{T}}(t)E_m(t)\right)^{-1}E_m^{\mathrm{T}}(t)Y_m(t) \tag{37}$$

According to Eq. (37), calculating $\hat{W}_m(t)$ requires calculating the inverse of $E_m(t)$. The dimension of $E_m(t)$ will increase, and the inverse computation will become time-consuming as iteration increases. To solve this problem, an iterative learning least square algorithm is derived.

Let us define $Q_m^{-1}(t) = E_m^{\mathrm{T}}(t)E_m(t)$, according to the definition of $E_m(t)$, we can get:

$Q_{m+1}^{-1}(t) = \sum_{i=0}^{m+1} \mathcal{E}_i(t)\mathcal{E}_i^{\mathrm{T}}(t) = \sum_{i=0}^{m} \mathcal{E}_i(t)\mathcal{E}_i^{\mathrm{T}}(t) +$ $\mathcal{E}_{m+1}(t)\mathcal{E}_{m+1}^{\mathrm{T}}(t)$ and namely:

$$Q_{m+1}^{-1}(t) = Q_m^{-1}(t) + \mathcal{E}_{m+1}(t)\mathcal{E}_{m+1}^{\mathrm{T}}(t) \tag{38}$$

Use Matrix inversion lemma, $(A^{-1} + HR^{-1}H^{\mathrm{T}})^{-1} = A - AH(H^{\mathrm{T}}AH + R)^{-1}H^{T}A$, let $A = Q_m(t)$, $H = \mathcal{E}_{m+1}(t)$, $R = 1$. we can get:

$$Q_{m+1}(t) = Q_m(t) - \frac{Q_m(t)\mathcal{E}_{m+1}(t)\mathcal{E}_{m+1}^{\mathrm{T}}(t)Q_m(t)}{1 + \mathcal{E}_{m+1}^{\mathrm{T}}(t)Q_m(t)\mathcal{E}_{m+1}(t)} \tag{39}$$

According to the definition of $Q_m^{-1}(t)$, we can compute $\hat{W}_m$ such that:

$$\begin{aligned} \hat{W}_{m+1} &= Q_{m+1}E_{m+1}^{\mathrm{T}}Y_{m+1} \\ &= Q_{m+1}\left[E_m^{\mathrm{T}}Y_m + \mathcal{E}_{m+1}y_{m+1}\right] \\ &= Q_{m+1}\left[Q_m^{-1}\hat{W}_m + \mathcal{E}_{m+1}y_{m+1}\right] \\ &= \hat{W}_m + Q_{m+1}\mathcal{E}_{m+1}\left[y_{m+1} - \mathcal{E}_{m+1}^{\mathrm{T}}\hat{W}_m\right] \end{aligned} \tag{40}$$

Define error $e_{m+1}(t) = y_{m+1}(t) - \mathcal{E}_{m+1}(t)\hat{W}_m(t)$

According to Eq. (39), we get:

$$\hat{W}_{m+1}(t) = \hat{W}_m(t) + \frac{Q_m(t)\mathcal{E}_{m+1}(t)}{1 + \mathcal{E}_{m+1}^{\mathrm{T}}(t)Q_m(t)\mathcal{E}_{m+1}(t)}e_{m+1}(t) \tag{41}$$

According to Eqs. (39) and (41), update laws for $Q_m(t)$ and $\hat{W}_m(t)$ indicate iterative learning least square algorithm of time-variant neural for network training.

Convergence analysis We analyze the convergence of iterative learning algorithm.

Theorem 1 *For system (33), the learning algorithm Eqs. (39)–(41) has the following properties:*

(i)

For $t = 0, 1, \ldots, N$ *and* $m = 0, 1, \ldots,$

$$\left\|\tilde{W}_{m+1}(t)\right\|^2 \leqslant \rho_m(t)\left\|\tilde{W}_m(t)\right\|^2 \tag{42}$$

$$\left\|\tilde{W}_m(t)\right\|^2 \leqslant \rho_0(t)\left\|\tilde{W}_0(t)\right\|^2 \tag{43}$$

where $\rho_m(t) = \lambda_{\max}(Q_m^{-1}(t))/\lambda_{\min}(Q_m^{-1}(t))$, *minimum eigenvalues and maximum eigenvalues of* $Q_m^{-1}(t)$ *are represented by* $\lambda_{\min}(Q_m^{-1}(t))$ *and* $\lambda_{\max}(Q_m^{-1}(t))$, *respectively.*

(ii) *For* $t = 0, 1, \ldots, N$,

$$\lim_{m\to\infty} e_{m+1}(t) = 0 \tag{44}$$

Proof By definition, $e_{m+1}(t) = \mathcal{E}_{m+1}^{\mathrm{T}}(t)\tilde{W}_m(t)$, according to Eqs. (39)–(41), we can get:

$$Q_{m+1}(t)Q_m^{-1}(t) = I - \frac{Q_m(t)\mathcal{E}_{m+1}(t)\mathcal{E}_{m+1}^{\mathrm{T}}(t)}{1 + \mathcal{E}_{m+1}^{\mathrm{T}}(t)Q_m(t)\mathcal{E}_{m+1}(t)} \tag{45}$$

$$\tilde{W}_{m+1}(t) = \left[I - \frac{Q_m(t)\mathcal{E}_{m+1}(t)\mathcal{E}_{m+1}^{\mathrm{T}}(t)}{1 + \mathcal{E}_{m+1}^{\mathrm{T}}(t)Q_m(t)\mathcal{E}_{m+1}(t)}\right]\tilde{W}_m(t) \tag{46}$$

Combine Eqs. (45) and (46), we can get:

$$Q_{m+1}^{-1}(t)\tilde{W}_{m+1}(t) = Q_m^{-1}(t)\tilde{W}_m(t) \tag{47}$$

Define $V_m(t) = \tilde{W}_m^{\mathrm{T}}(t)Q_m^{-1}(t)\tilde{W}_m(t)$, use Eq. (45), we can get:

$$V_{m+1}(t) - V_m(t) = -\frac{\tilde{W}_m^{\mathrm{T}}(t)\mathcal{E}_{m+1}(t)\mathcal{E}_{m+1}^{\mathrm{T}}(t)\tilde{W}_m(t)}{1 + \mathcal{E}_{m+1}^{\mathrm{T}}(t)Q_m(t)\mathcal{E}_{m+1}(t)} \tag{48}$$

Therefore, $V_m(t)$ are non-increasing and nonnegative for m, namely

$$V_{m+1}(t) \leqslant V_m(t) \tag{49}$$

According to Eq. (38), we can get:

$$\lambda_{\min}(Q_{m+1}^{-1}(t)) \geqslant \lambda_{\min}(Q_m^{-1}(t)) \geqslant \lambda_{\min}(Q_0^{-1}(t)) \tag{50}$$

Combine Eq. (49), we can get:

$$\lambda_{\min}(Q_m^{-1}(t))\left\|\tilde{W}_{m+1}(t)\right\|^2 \leqslant \lambda_{\max}(Q_m^{-1}(t))\left\|\tilde{W}_m(t)\right\|^2 \tag{51}$$

Therefore:

$$\left\|\tilde{W}_{m+1}(t)\right\|^2 \leqslant \rho_m(t)\left\|\tilde{W}_m(t)\right\|^2 \tag{52}$$

Apparently there is $V_m(t) \leqslant V_0(t)$, combine Eq. (49), we can get:

$$\lambda_{\min}(Q_0^{-1}(t))\left\|\tilde{W}_m(t)\right\|^2 \leqslant \lambda_{\max}(Q_0^{-1}(t))\left\|\tilde{W}_0(t)\right\|^2 \tag{53}$$

This ensures that $\left\|\tilde{W}_m(t)\right\|^2$ is uniformly bounded. The property (i) is proved. □

Proof According to Eq. (48), we can get:

$$\frac{e_{m+1}^2(t)}{1 + \mathcal{E}_{m+1}^{\mathrm{T}}(t)Q_m(t)\mathcal{E}_{m+1}(t)} = V_m(t) - V_{m+1}(t) \tag{54}$$

Sum the above formula from 0 to m, we get:

$$\sum_{i=0}^{m}\frac{e_{i+1}^2(t)}{1 + \mathcal{E}_{i+1}^{\mathrm{T}}(t)Q_i(t)\mathcal{E}_{i+1}(t)} = V_0(t) - V_{m+1}(t) \leqslant V_0(t) \tag{55}$$

Because $V_0(t)$ is bounded, therefore:

$$\lim_{m\to\infty}\frac{e_{m+1}^2(t)}{1 + \mathcal{E}_{m+1}^{\mathrm{T}}(t)Q_m(t)\mathcal{E}_{m+1}(t)} = 0 \tag{56}$$

Use $\lambda_{\max}(Q_{m+1}(t)) \leqslant \lambda_{\max}(Q_m(t))$, we can get:

$$\mathcal{E}_{m+1}^{\mathrm{T}}(t)Q_m(t)\mathcal{E}_{m+1}(t) \leqslant \lambda_{\max}(Q_0(t))\mathcal{E}_{m+1}^{\mathrm{T}}(t)\mathcal{E}_{m+1}(t) \tag{57}$$

According to Eq. (56), we can get:

$$\lim_{m\to\infty}\frac{e_{m+1}(t)}{\sqrt{1 + \lambda_{\max}(Q_0(t))\mathcal{E}_{m+1}^{\mathrm{T}}(t)\mathcal{E}_{m+1}(t)}} = 0 \tag{58}$$

The input $x_m(t)$ is bounded; then, $\mathcal{E}_{m+1}(t)$ is bounded. In Eq. (58), the denominator term is bounded and nonzero; then, we can get:

$$\lim_{m\to\infty} e_{m+1}(t) = 0, t = 0, 1, \ldots, N \tag{59}$$

□

This ensures the uniform convergence of the estimation error and proves the property (ii). Therefore, the learning algorithm we give ensures the boundedness of time-variant weights. The estimation error on a finite time interval can asymptotically converge to zero.

Acknowledgements This work was supported by the Fundamental Research Funds for the Central Universities (Grant No. 2020-JBZD004).

Declarations

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

References

1. Yule GU (1927) On a method of investigating periodicities in disturbed series, with special reference to Wolfer's sunspot numbers. Philos Trans R Soc A 226:267–298
2. Box GEP, Pierce DA (1970) Distribution of residual in autoregressive-integrated moving average time series. J Am Stat Assoc 65:1509–1526
3. Chen S, Wang XX, Harris CJ (2007) NARX-based nonlinear system identification using orthogonal least squares basis hunting. IEEE Trans Control Syst Technol 16(1):78–84
4. Bouchachia A, Bouchachia S (2008) Ensemble learning for time series prediction. NDES
5. Frigola R, Rasmussen CE (2013) Integrated pre-processing for Bayesian nonlinear system identification with gaussian processes. In: IEEE CDC
6. Rumelhart DE, Hinton GE, Williams RJ (1986) Learning representations by back-propagating errors. Nature 323(6088):533–536
7. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
8. Gers FA, Schmidhuber J, Cummins F (2000) Learning to forget: continual prediction with LSTM. Neural Comput 12(10):2451–2471
9. Gers FA, Eck D, Schmidhuber J (2001) Applying LSTM to time series predictable through time-window approaches. In: ICANN, pp 669–676
10. Khandelwal U, He H, Qi P, Jurafsky D (2018) Sharp nearby, fuzzy far away: how neural language models use context. ACL, pp 284–294
11. Mariet Z, Kuznetsov V (2019) Foundations of sequence-to-sequence modeling for time series. CoRR arXiv:1805.03714
12. Vaswani A, Shazeer N, Parmar N, Uszkoreit J, Jones L, Gomez A, Kaiser L, Polosukhin I (2017) Attention is all you need. In: NIPS
13. Guo M, Zhang Y, Liu T (2019) Gaussian transformer: a lightweight approach for natural language inference. AAAI 33:6489–6496
14. Im J, Cho S (2017) Distance-based self-attention network for natural language inference. In: CoRR arXiv:1805.03714
15. Dabrowski JJ, Zhang Y, Rahman A ForecastNet: A time-variant deep feed-forward neural network architecture for multi-step-ahead time-series forecasting. In: ICONIP
16. Qin Y, Song D, Chen H, Cheng W, Jiang G, Cottrell G (2017) A dual-stage attention-based recurrent neural network for time series prediction. In: IJCAI
17. Lai G, Chang WC, Yang Y, Liu H (2018) Modeling long- and short-term temporal patterns with deep neural networks. ACM
18. Taieb SB, Atiya AF (2015) A bias and variance analysis for multistep-ahead time series forecasting. IEEE Trans Neural Netw Learn Syst 27:62–76
19. Taieb SB, Bontempi G, Atiya AF, Sorjamaa A (2012) 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
20. Salinas D, Flunkert V, Gasthaus J, Januschowski T (2020) DeepAR: probabilistic forecasting with autoregressive recurrent networks. Int J Forecast 36(3):1181–1191
21. Rangapuram SS, Seeger MW, Gasthaus J, Stella L, Wang Y, Januschowski T (2018) Deep state space models for time series forecasting. In: NIPS, pp 7785–7794
22. Wang Y, Smola A, Maddix D, Gasthaus J, Foster D, Januschowski T (2019) Deep factors for forecasting. In: ICML, pp 6607–6617
23. Bengio Y, Simard P, Frasconi P (1994) Learning long-term dependencies with gradient descent is difficult. IEEE Trans Neural Netw 2:157–166
24. Sutskever I, Vinyals O, Le QV (2014) Sequence to sequence learning with neural networks. NIPS, pp 3104–3112
25. Yu R, Zheng S, Anandkumar A, Yue Y (2017) Long-term forecasting using higher order tensor RNNs. CoRR arXiv:1711.00073
26. Wen R, Torkkola K, Narayanaswamy B, Madeka D (2017) A multi-horizon quantile recurrent forecaster. NIPS
27. Guen VL, Thome N (2019) Shape and time distortion loss for training deep time series forecasting models. In: NIPS, pp 4189–4201
28. Laptev N, Yosinski J, Li LE, Smyl S (2017) Time-series extreme event forecasting with neural networks at uber. ICML 34:1–5
29. Xiong H, He Z, Wu H, Wang H (2018) Modeling coherence for discourse neural machine translation. In: AAA I, pp 7338–7345
30. Yang B, Wang L, Wong DF, Chao LS, Tu Z (2019) Convolutional self-attention network. CoRR arXiv:1810.13320
31. Shaw P, Uszkoreit J, Vaswani A (2018) Self-attention with relative position representations. In: NAACL
32. Zhang J, Luan H, Sun M, Zhai F, Xu J, Zhang M, Liu Y (2018) Improving the transformer translation model with document-level context. In: EMNLP
33. Im J, Cho S (2017) Distance-based self-attention network for natural language inference. CoRR arXiv:1712.02047
34. Yang B, Tu Z, Wong DF, Meng F, Chao LS, Zhang T (2018) Modeling localness for self-attention networks. CoRR arXiv:1810.10182
35. Devlin J, Chang MW, Lee K, Toutanova K (2019) BERT: pre-training of deep bidirectional transformers for language understanding. In: NAACL, pp 4171–4186
36. Kang WC, Mcauley J (2018) Self-attentive sequential recommendation. In: ICDM
37. Sun F, Liu J, Wu J, Pei C, Lin X, Ou W, Jiang P (2019) BERT4Rec: sequential recommendation with bidirectional encoder representations from transformer. CoRR arXiv:1904.06690
38. Zhang S, Tay Y, Yao L, Sun A (2018) Next item recommendation with self-attention. CoRR arXiv:1808.06414
39. Chen Q, Zhao H, Li W, Huang P, Ou W (2019) Behavior sequence transformer for E-commerce recommendation in Alibaba. In: KDD
40. Liu PJ, Saleh M, Pot E, Goodrich B, Sepassi R, Kaiser L, Shazeer N (2018) Generating wikipedia by summarizing long sequences. In: ICLR
41. Hoang A, Bosselut A, Celikyilmaz A, Choi Y (2019) Efficient adaptation of pretrained transformers for abstractive summarization. CoRR arXiv:1906.00138
42. Subramanian S, Li R, Pilault J, Pal C (2019) On extractive and abstractive neural document summarization with transformer language models. CoRR arxiv:1909.03186
43. Zhang X, Meng K, Liu G (2019) Hie-transformer: a hierarchical hybrid transformer for abstractive article summarization. In: ICONIP, pp 248–258

44. Wu N, Green B, Ben X, O'Banion S (2020) Deep transformer models for time series forecasting: the influenza prevalence case. CoRR arXiv:2001.08317
45. Huang S, Wang D, Wu X, Tang A (2019) DSANet: dual self-attention network for multivariate time series forecasting. ACM, pp 2129–2132
46. Song H, Rajan D, Thiagarajan JJ, Spanias A (2018) Attend and diagnose: clinical time series analysis using attention models. In: AAAI
47. Yang B, Tu Z, Wong DF, Meng F, Chao LS, Zhang T (2018c) Modeling localness for self-attention networks. In: EMNLP, pp 4449–4458
48. Oppenheim AV, Schafer RW, Buck JR (1999) Pearson Education Signal Processing Series, Discrete-Time Signal Processing (2nd ed.)
49. Hyndman R Time series data library. https://robjhyndman.com/tsdl/
50. Guin A (2006) Travel time prediction using a seasonal autoregressive integrated moving average time series model. In: ITSC, pp 493–498
51. Fildes R (1992) Bayesian forecasting and dynamic models. Int J Forecast 8:635–637
52. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. In: ICLR
53. Bai S, Kolter JZ, Koltun V (2018) An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR arxiv:1803.01271
54. Hyndman RJ, Koehler AB (2006) Another look at measures of forecast accuracy. Int J Forecast 22(4):679–688
55. Hyndman R, Athanasopoulos G (2018) Forecasting: principles and practice. OTexts
56. Makridakis S, Spiliotis E, Assimakopoulos V (2018) The M4 competition: results, findings, conclusion and way forward. Int J Forecast 34(4):802–808
57. Sun M (2009) Iterative learning neurocomputing. In: WNIS, pp 158–161