Expert Systems With Applications 201 (2022) 117121

Contents lists available at ScienceDirect

Expert Systems With Applications

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

Pre-SMATS: A multi-task learning based prediction model for small multi-stage seasonal time series

Shiling Wu, Dunlu Peng *

School of Optical-Electrical and Computer Engineering, University of Shanghai for Science and Technology, 200093, Shanghai, China

ARTICLE INFO

Keywords:
Small seasonal time series
Multi-task learning
Data prediction
Neural networks
Multi-stage
Feature enhancement

ABSTRACT

Learning on time series, especially on the small seasonal time series, has a wide range of practical applications. In this paper, to improve the learning effect on small seasonal time series, we extract the implicit information from the original series to enhance the data features and utilize stages to convey seasonality. Considering that many seasonal time series have implied stage characteristics, we propose Pre-SMATS, a multi-task learning based prediction model for small multi-stage seasonal time series. The model consists of three components: a feature extractor, a sub-task classifier and a main task predictor. The feature extractor is used to learn the feature vectors of the input data, the sub-task classifier is employed for stage classification and the main task predictor is applied for the final prediction. After extracting features of the input data, the extracted vectors are passed into the sub-task classifier and the main task predictor, respectively. Having learned the stage vector (i.e. the embedding of stage characteristics), the sub-task classifier outputs it to the main task, which concatenates the stage vector with the feature vector extracted by the feature extractor, to improve the prediction accuracy of the model. In order to verify our proposed model, Pre-SMATS, we conducted extensive experiments on two datasets with small time series, one is Chinese civil aviation passenger traffic (CCAPT) (2013–2019), a multi-stage seasonal time series and the other is IC board production furnace temperature curve (FTC), a general multi-stage time series. The experimental results show that our proposed model is superior to the baseline models in generalization capability as well as the accuracy of prediction on both two datasets. Specifically, compared with the best-performing baseline neural network model, the five error metrics (MSE, RMSE, MAE, MAPE and MSLE) of Pre-SMATS on CCAPT are reduced by 34%, 19%, 24%, 23% and 39% respectively, and on the FTC, they dropped by 42%, 25%, 16%, 18% and 43%.

1. Introduction

Time series is a series of data that changes over time and is widely used in real-world applications, such as virus propagation (Istaiteh, Owais, Al-Madi, & Abu-Soud, 2020), business operations forecasting (du Jardin, 2021), stock forecasting (Shankar, Arora, Kaushal, & Singh, 2019) and traffic management (Tang, Liu, Zou, Zhang, & Wang, 2017; Yuan & Li, 2021). In many application scenarios, it is necessary to extract information from time series in order to accurately predict the future. Currently, the research of time series can be generally divided into prediction (Hua et al., 2019; Yan & Ouyang, 2018) and classification (Fawaz, Forestier, Weber, Idoumghar, & Muller, 2019; Fawaz et al., 2020), which may also involve the detection and processing of missing values and outliers (Li, Zhang, Wang, & Ran, 2018; Lim, Kim, Park, & Kwon, 2021; Wei, 2006). In this paper, we focus on the prediction of time series without missing values, particularly, seasonal time series. *Seasonal time series*, which has *seasonal characteristics*, is a kind of time-series data that appears repeatedly in a fixed time period, and the minimum time period of this repetitive phenomenon is called *seasonal period* (Hyndman, 2011; Wei, 2006).

So far, there are three main approaches for predicting seasonal time series: statistical models (Lima, Gonçalves, & Costa, 2019; Noureen, Atique, Roy, & Bayne, 2019), traditional machine learning (Martínez, Frías, Pérez-Godoy, & Rivera, 2018; Zhang & Wang, 2018), and neural network (Ashour & Abbas, 2018; Tang et al., 2017; Zhang, Yuan, et al., 2020). Statistical models usually use decomposition methods to extract seasonal characteristics from time series and add them as independent terms into the model. Noureen et al. (2019) leveraged SARIMA, a model that incorporate seasonal features to ARIMA, to predict the seasonal agricultural load. Lima et al. (2019) compared the performance of the additive and the multiplicative Holt-Winters Exponential Smoothing models by forecasting economic time series with trends and seasonal patterns. The difference between the two is whether the seasonal

* Corresponding author.
E-mail addresses: wusi0909@126.com (S. Wu), pengdl@usst.edu.cn (D. Peng).

https://doi.org/10.1016/j.eswa.2022.117121
Received 22 July 2021; Received in revised form 13 February 2022; Accepted 28 March 2022
Available online 12 April 2022

components are involved in the operation in the form of addition or multiplication.

Unlike statistical models, there is no general consensus on the treatment of seasonal time series by traditional machine learning and neural network. For example, Martínez et al. (2018) used different learners, each of which is trained and predicted for a specific season only, to achieve the prediction of a complete seasonal time series by combining learners. Zhang and Wang (2018) employed singular spectrum analysis techniques to identify and extract trend and seasonal components in the electricity load series, and applies ARIMA to model the linear variables and SVM to model the nonlinear variables, combining the advantages of both models to improve the prediction performance. In the field of neural networks, a fuzzy neural network was proposed to predict traffic speed, and trigonometric functions were also employed to fit the seasonal characteristics (Tang et al., 2017), and for the predication of PM2.5, in (Zhang, Yuan, et al., 2020), a method based on an attention mechanism was presented to give different weights to different weather data without additionally dealing with the seasonal characteristics.

Small time series (STS) refers to the time series with relative short or small sample size. Learning on small time series, especially on small seasonal time series, has a wide range of practical applications. However, training on small datasets is often difficult to obtain good results (Hayashi, Tanimoto, & Kashima, 2019), although deep learning has achieved great success by its excellent feature extraction and feature combination ability in recent years (Zhou, Zhao, Wang, & Liu, 2018). In general, deep learning models cannot be directly generalized to small time series learning (Keshari, Ghosh, Chhabra, Vatsa, & Singh, 2020). To date, some studies have focused on small sample learning. The literature (Moreno-Barea, Jerez, & Franco, 2020) employs different data enhancement techniques to improve the classification accuracy of small datasets. Molina, Asencio-Cortés, Riquelme, and Martínez-Álvarez (2020) used transfer learning to improve the image classification performance of convolutional neural networks on small sample datasets. In literature (Fang et al., 2020), a decision framework combining depth Gaussian process and feedback control method is proposed to solve the problem of insufficient data in the field of automatic driving.

Different from these methods, this paper attempts to extract the implicit information of the original STS to enhance series features and express the seasonal characteristics of STS so that improve the learning effect on STS. Inspired by idea of multi-task learning, we propose Pre-SMATS, a new neural network model, to accomplish prediction task on small multi-stage seasonal time series. After defining the stage characteristics, the series is labeled by observing the stage characteristics within each cycle of the seasonal time series, and then get all stage labels of the whole series. With the help of these stage labels, the model can further mine the stage characteristics, thus realize the feature enhancement and seasonal expression of the original series, and the multi-task learning model is constructed to implement the stage prediction and the final prediction simultaneously.

The main contributions of this study are summarized as follows:

- A multi-task learning based prediction model, Pre-SMATS, for predicting small multi-stage seasonal time series is presented, in which one extracted feature vector is input into the sub-task classifier to generate the stage vector, and the vector is output to the main task and connected with the other feature vector calculated by the feature extractor to conduct the final prediction. By enhancing data features, the model can reduce the impact of insufficient data and improve prediction accuracy.
- We propose a new approach for processing seasonal characteristics. Different from the existing methods, such as differencing and decomposing the data to eliminate the seasonal component or using periodic function to fit seasonal characteristics, this work introduces a method for labeling seasonal time series with stages, and in this way, the task of acquiring seasonal characteristics is transformed into acquiring stage characteristics, which expresses seasonal characteristics in a finer granularity.
- To verify our model, extensive experiments were carried out on two real small time series datasets, a seasonal time series – the Chinese civil aviation passenger traffic (2013–2019) and a general multi-stage time series – the IC board production furnace temperature curve. The experimental results demonstrate that our proposed model does have an improved effect on the STS. Specifically, Pre-SMATS not only has excellent generalization capability which is valid for both small multi-stage seasonal time series and small general multi-stage time series, but also performs better in prediction than the baseline models, such as ARIMA (Box, Jenkins, & Reinsel, 1970), SARIMA (Montgomery, Johnson, & Gardiner, 1990), TCN (Bai, Kolter, & Koltun, 2018), LSTM (Hochreiter & Schmidhuber, 1997), Bi-LSTM (Schuster & Paliwal, 1997), LSTM+Attention and Bi-LSTM+Attention.

The rest of our paper is structured as follows. In Section 2, we describe existing time series predicting methods and briefly introduce the basis of our model proposed, and in Section 3, after defining the problem, we present the idea and framework of the proposed Pre-SMATS model. Section 4 presents the experiments and the analysis of experimental results. Finally, in Section 5, we draw conclusions and introduce our future work.

2. Related work

In this section, we describe some representative existing models for predicting time series and the foundations of our proposed models, such as ARIMA, SARIMA, TCN, LSTM, Bi-LSTM, Attention mechanism (Vaswani et al., 2017), Generic additive network (Lopez-Martin, Carro, & Sanchez-Esguevillas, 2019), Multi-task learning (Caruana, 1997), etc.

2.1. ARIMA

ARIMA, the autoregressive integrated moving average model, as a common statistical model used for predicting time series, is usually represented by a triple $ARIMA(p,d,q)$, where p is the number of autoregressive terms, q is the number of sliding average terms and d is the number of difference to make the time series stationary.

The specific modeling process of ARIMA is given in Fig. 1, which displays the model requires the original input data to be stationary, at least after being differenced. After the data is entered, it is tested for stationarity at first, and if it is non-stationary then the difference operation is performed until it is stationary. Then, the auto-regressive and moving average model (ARMA) is applied to model the data. The model is validated by a white noise test to determine if the modeling is successful. If the modeling is unsuccessful, the model continues to adjust its parameters till it is successful. Although, as a good tool for modeling stationary data, ARIMA has been widely used in tourism (Ismail, 2019), industry (Fan et al., 2020) and fisheries (Selvaraj, Arunachalam, Coronado-Franco, Romero-Orjuela, & Ramírez-Yara, 2020), but it does not work well for solving non-linear fluctuating data. In addition, when the data is seasonal, it is difficult to make it stationary by using ARIMA, which reduces the interpretability of the model. Another disadvantage of ARIMA is that it is not suitable for long-term prediction.

2.2. SARIMA

Seasonal autoregressive integrated moving average (SARIMA) is a modification of ARIMA and is a predicting method for time series with seasonal characteristics. Corresponding to ARIMA, SARIMA is generally denoted as $ARIMA(p,d,q)(P,D,Q)_S$, where P,D,Q stands for p,d,q with the additional seasonal characteristics, and S is the seasonal period. The modeling process of SARIMA is similar to that of ARIMA, as shown in Fig. 1.

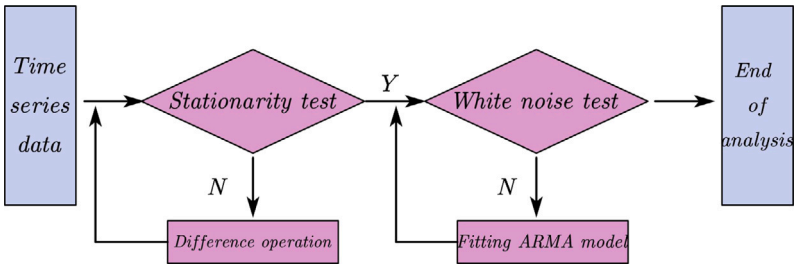


Fig. 1. The modeling process of ARIMA.

SARIMA has been applied to predict the seasonal time series, such as air conditions (Zhang, Yuan, et al., 2020), the number of flights passengers (Carmona-Benítez & Nieto, 2020), and regional rainfall (Nwokike, Offorha, Obubu, Ugoala, & Ukomah, 2020). It divides non-stationary factors in seasonal time series into seasonal and trending factors, and then respectively transform them into stationary by differencing for prediction, which makes up for the shortcomings of ARIMA for predicting seasonal time series. However, like ARIMA, SARIMA is unable to capture the non-linear relationships in the series and can only make short-term predictions.

2.3. TCN

The temporal convolutional network (TCN) was originally proposed by Lea et al. for solving the problem of segmenting and detecting actions in video (Lea, Flynn, Vidal, Reiter, & Hager, 2017). The main features of TCN are: 1) the convolution in the architecture is causal; 2) like RNN, the network can take a sequence of arbitrary length and map it to an output sequence of the same length. In order to achieve the above functions it consists of a one-dimensional fully convolutional network and causal convolutions (Bai et al., 2018). TCNs are faster to train and can avoid gradient explosion or gradient disappearance that often occurs in RNNs. In recent years, TCN has been widely used in addressing time series issues. For example, the literature (Hewage et al., 2020) exploited TCN to forecast the weather and achieved better results than the LSTM. In literature (Lin, Xu, Wu, Richardson, & Bernal, 2019), a hierarchical attention-based temporal convolutional network model was proposed for medical time series classification, and it was demonstrated that the model has better robustness to noise compared to an attention-based LSTM. Yan, Mu, Wang, Ranjan, and Zomaya (2020) proposed the ensemble empirical mode decomposition-temporal convolutional network hybrid approach for the advance prediction of El Niño-Southern Oscillation.

2.4. LSTM

Long short-term memory (LSTM) is a neural network proposed by Hochreiter and Schmidhuber (1997) to solve the long-standing problem of vanishing gradient and exploding gradient in RNNs. In addition to alleviating the problem of vanishing gradient and exploding gradient from the design of neural network, based on the back propagation algorithm, Abuqaddom, Mahafzah, and Faris (2021) designed a new loss reduction algorithm to solve the problem of vanishing gradient. As shown in Fig. 2, LSTM uses logic gates (input gate, output gate and forgetting gate) to make it have the function of memorizing time series, so that it can selectively process data. The functions of the logic gates are expressed as follows

$$f_t = \sigma\left(\mathbf{W_f} \cdot \left[\mathbf{h_{t-1}}, \mathbf{x_t}\right] + \mathbf{b_f}\right) \tag{1}$$

$$i_t = \sigma\left(\mathbf{W_i} \cdot \left[\mathbf{h_{t-1}}, \mathbf{x_t}\right] + \mathbf{b_i}\right) \tag{2}$$

$$\widetilde{\mathbf{C}}_\mathbf{t} = \tanh\left(\mathbf{W_c} \cdot \left[\mathbf{h_{t-1}}, \mathbf{x_t}\right] + \mathbf{b_c}\right) \tag{3}$$

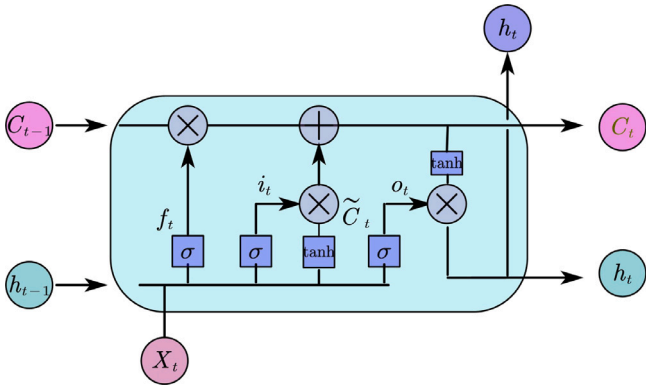


Fig. 2. The structure of LSTM neural network.

$$\mathbf{C_t} = f_t \circ \mathbf{C_{t-1}} + i_t \circ \widetilde{\mathbf{C}}_\mathbf{t} \tag{4}$$

$$o_t = \sigma\left(\mathbf{W_o} \cdot \left[\mathbf{h_{t-1}}, \mathbf{x_t}\right] + \mathbf{b_o}\right) \tag{5}$$

$$\mathbf{h_t} = o_t \circ \tanh\left(\mathbf{C_t}\right) \tag{6}$$

where $\mathbf{x}$ is the input vector, f stands for forget gate outputs, i stands for input gate outputs, $\widetilde{\mathbf{C}}$ stands for candidate cell states, $\mathbf{C}$ describes cell states, o describes output gate outputs, $\mathbf{h}$ stands for hidden states, t denotes the tth moment, $\mathbf{W_f}$, $\mathbf{W_i}$, $\mathbf{W_c}$, $\mathbf{W_c}$, $\mathbf{W_o}$ and $\mathbf{b_f}$, $\mathbf{b_i}$, $\mathbf{b_c}$, $\mathbf{b_o}$ denote the corresponding weight matrices and bias respectively, σ and tanh are the *Sigmoid* function and hyperbolic tangent activation function correspondingly, and $\circ$ means Hadamard product.

Owing to its excellent ability to capture data patterns, which can take advantage of obtaining correlation features such as trends, seasonality, autocorrelation and noise, LSTM are often used as a powerful tool for time series prediction (Gautam, 2021). Gautam (2021) combined transfer learning with LSTM to predict the number of COVID-19 cases and deaths. In the work of (Zang et al., 2020), LSTM was used for short-term solar radiation prediction under different seasonal and sky conditions during the year, and Du, Zhou, Guo, Guo, and Wang (2021) worked with LSTM to predict the daily water demand of cities, which is also an application case for seasonal time series.

2.5. Bi-LSTM

Bi-directional long short-term memory (Bi-LSTM) consists of two layers of LSTMs. The first layer applies the LSTM to the input sequence (i.e., the forward layer). The second layer takes the opposite form of the input sequence into the LSTM model (i.e., the backward layer). It combines the advantages of the LSTM model in better capturing longer distance dependencies and improves on the shortcomings of the LSTM in not capturing contextual relationships (El-Sappagh, Abuhmed, Islam,

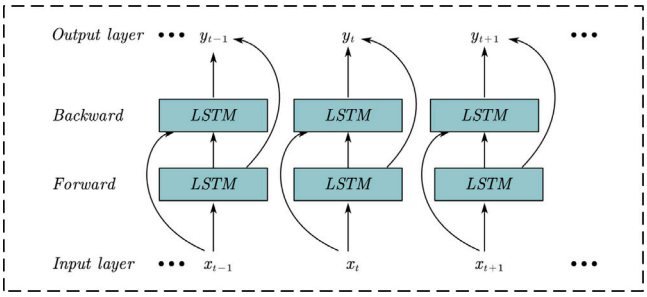


Fig. 3. The structure of Bi-LSTM neural network.

& Kwak, 2020; Jin et al., 2020; Livieris, Pintelas, & Pintelas, 2020), as shown in Fig. 3.

Many studies have shown that the Bi-LSTM does have higher accuracy than the LSTM in quite a few areas. The literature (Zhang, Liu, et al., 2020) used Bi-LSTM for air quality prediction and obtained a higher accuracy rate compared to LSTM. Meanwhile, Bi-LSTM is also more accurate than LSTM in predicting tourism demand (Kulshrestha, Krishnaswamy, & Sharma, 2020), PM2.5 concentration (Zhang, Zhang, et al., 2020), and soil water evaporation (Yin, Deng, Ines, Wu, & Rasu, 2020).

2.6. Attention mechanism

Attention mechanisms are extensively applied in various areas of deep learning (Bahdanau, Cho, & Bengio, 2014; Mnih, Heess, Graves, & Kavukcuoglu, 2014; Yin, Schütze, Xiang, & Zhou, 2016), its core idea is to focus attention on the key points in a mass of information, selecting the critical information and ignoring other insignificant information. In the field of time series forecasting, Qin et al. (2017) proposed a novel dual-stage attention-based recurrent neural network (DA-RNN), and the experiments on the SML 2010 dataset and the NASDAQ 100 Stock dataset suggested that DA-RNN outperform RNN networks with single attention mechanism. Literature (Liang, Ke, Zhang, Yi, & Zheng, 2018) presented a new attention-based multi-level network for predicting geo-sensory time series based on heterogeneous data from multiple domains, and a simply attend and diagnose architecture has been designed for processing clinical time series based on the idea of multi-task learning combined with multi-headed attention (Song, Rajan, Thiagarajan, & Spanias, 2018).

2.7. Generic additive network

The generic additive network (gaNet) was proposed by Lopez-Martin et al. (2019) for supervised regression. The building elements of the architecture are learning blocks, which may consist of a series of fully connected layers, more complex recurrent and convolutional layers, and are trained end-to-end using stochastic gradient descent. Experimental results on network traffic dataset show that this method has advantages in both prediction performance and training prediction time. Subsequently, gaNet-C was proposed for supervised classification (Lopez-Martin, Carro, & Sanchez-Esguevillas, 2020). Lopez-Martin, Sanchez-Esguevillas, Hernandez-Callejo, Arribas, and Carro (2021) presented a constrained weighted quantile loss for probabilistic electric-load forecasting and the best results were obtained when experimenting with additive ensemble neural network as the base model.

2.8. Multi-task learning

Multi-task learning allows multiple related tasks to be processed simultaneously and exploits the correlation between tasks to improve the learning performance (Ma & Tan, 2020). Multiple tasks share a single structure, and the parameters of the structure are affected by all tasks when they are optimized, which makes the structure equivalent to fusing all tasks when all of them converge. Therefore, in general, the generalization capability of multi-task learning is better than that of a single task learning.

Multi-task learning has been applied to many fields. In literature (Zeng, Mao, Peng, & Yi, 2019), multi-task learning was used to accomplish the task of audio classification and achieved higher accuracy than the single task learning. In literature (Wen et al., 2020), a multi-task learning based recommendation system was proposed to solve the sample selection bias and data sparsity issues. Alom, Aspiras, Taha, Bowen, and Asari (2020) used multi-task learning to process the mitotic cell detection task, which was considered as a combination of sub-tasks including cell segmentation, detection and classification, and experimental results showed that the method could meet the requirements of clinical practice.

In our work, we first analyze the stage characteristics of the small seasonal time series within each cycle and annotate the time series with stages. The method not only describes seasonal characteristics at a finer granularity, but also compensates for the disadvantage of insufficient data by feature enhancement. In order to obtain the stage characteristics, we need to classify the series into stages. This task needs to be completed before the final prediction can be achieved. Inspired by the idea of multi-task learning, we attempt to establish a model that can accomplish these two tasks simultaneously.

3. Pre-SMATS — Our model

In this section, we first formally describe the small multi-stage seasonal time series prediction task and then detail our model, Pre-SMATS.

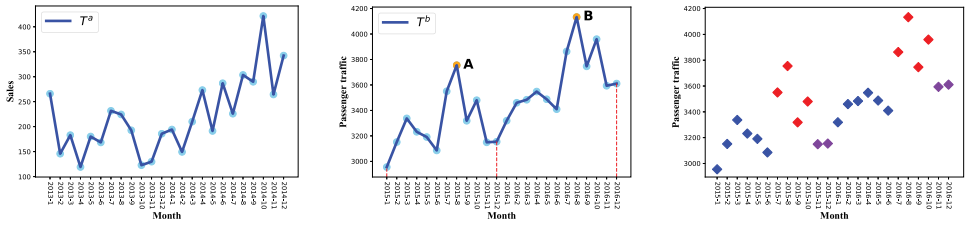
3.1. Problem definition

Before formulating our problem, we need to define some concepts used in the description.

Definition 1. If a time series $S = \left\{ s_{t_1}, s_{t_2}, \ldots, s_{t_n} \right\}$, s_{t_i} is the value of the time series S at time point t_i, the length of time between s_{t_i} and $s_{t_{i+1}}$ is called a time interval. The sub-series $S^a = \left\{ s_{t_{i+1}}, \ldots, s_{t_{i+p}} \right\}$, is the sub-series generated by S at a period of time p ($p > 1$) and the sub-series $S^b = \left\{ s_{t_{i+p+1}}, \ldots, s_{t_{i+(2*p)}} \right\}$ is the sub-series of S generated in the post-p time period of S^a. If S^a and S^b have similar shape, then, S has **seasonal characteristics**, and S is called **a seasonal time series** and p is the **seasonal period**.

Fig. 4(a) depicts T^a, a time series of monthly sales of a product, in which is hard to find sub-series with similar trends, indicating that it is nonseasonal. Fig. 4(b) describes, T^b, the time series of passenger traffic of a certain civil aviation from 01-01-2015 to 31-12-2016. If T^b is divided by a period of 12 months, for example, 01-01-2015~ 31-12-2015 and 01-01-2016~ 31-12-2016, it is found that the variation trend of the two sub-series is very similar, and they both have a peak point, i.e. Point A and Point B, respectively, implying that the passenger traffic is the highest in August every year. This suggests the T^b is a seasonal time series. Let us take a close look at the scatter plot of the time series in Fig. 4(c), we can see that points of the same color are clustered more closely in the same cycle, and the data points have the same behavior at the same position in the next cycle. This inspired us that the series can be divided into stages according to the degree of dispersion between data points contained in the cycle, in other words, the seasonal time series can be expressed through the cycle of stages.

In statistics, the metrics used to evaluate the degree of data dispersion is coefficient of variation c_v, which is measured with the quotient of the standard deviation (σ) and the average value (μ), i.e., $c_v = \frac{\sigma}{\mu}$. Generally, the coefficient of variation increases with the degree of data dispersion.



(a) Nonseasonal time series (b) Seasonal time series with a period of 12 (c) The scatter plot of Figure 4(b)

Fig. 4. Example of seasonal time series.

Definition 2. Given a time series $S = \left\{ s_{t_1}, s_{t_2}, \ldots, s_{t_n} \right\}$, S_w is a sliding window with an initial size of $w(w \geq 2)$ and starts with $s_{t_j}(j \geq 1)$. S_w gradually expands in steps of $k(k \geq 1)$, and after expanding $i(i \geq 1)$ times, the set of data contained in S_w is A_i with coefficient of variation $c_{v_{A_i}}$. If $\Delta(c_{v_{A_i}}, c_{v_{A_{i-1}}}) \geq \beta$ (β is the given threshold), then A_i is a new **stage** of S, and $s_{t_{j+(w+k*(i-1))+1}}$ is the starting point of the next stage. In this way, S can be divided into multiple stages and denoted as $S = \left\{ \underbrace{s_{t_1}, s_{t_2}, \ldots, s_{t_u}}_{1}, \underbrace{s_{t_{u+1}}, \ldots, s_{t_z}}_{2}, \underbrace{\cdots}_{\ldots}, \underbrace{s_{t_j}, \ldots, s_{t_{j+(w+k*(i-1))}}}_{l}, \underbrace{\cdots}_{\ldots}, \underbrace{s_{t_u}, \ldots, s_{t_n}}_{m} \right\}$, which means S has **stage characteristics** and is defined as a **multi-stage time series**.

Definition 3. Suppose $S = \left\{ s_{t_1}, s_{t_2}, \ldots, s_{t_n} \right\}$ is a seasonal time series with a period of p, and D is the set of sub-series of each individual cycle of S. Let $\forall S^a, S^b \in D$, if both S^a and S^b are **multi-stage time series**, we call S as a **multi-stage seasonal time series**.

Definition 3 tells us that, besides the seasonal characteristics, multi-stage seasonal time series also have stage characteristics. Therefore, in the analysis of small multi-stage seasonal time series, by the utilization of the stage labeling information, we can not only enhance the features of the series, but also express the seasonal time series in a finer granularity, which is conducive to improving the accuracy of prediction. The key to achieve this goal is how to obtain the embedding of stage characteristics of the small multi-stage seasonal time series.

In order to obtain the embedding of stage characteristics, we need a feature extractor H to transform the input series into feature vectors. Furthermore, in order to obtain the stage vector of the feature vector $\mathbf{f}$, a classifier G_p is constructed, which is combined with the learning block and the last fully connected (FC) layer with a $Softmax$ function to predict the stage to which the input data belongs at the next moment. With the assistance of G_p, the vector $\tilde{\mathbf{f}}$ is obtained which associates the feature vector of the input data with the stage, i.e., $\tilde{\mathbf{f}}$ is the stage vector of the input data $\bar{S}$, which is the embedding of stage characteristics. After extracting the stage vector $\tilde{\mathbf{f}}$, we conduct the secondary feature extraction on the original time series by regarding it as an ordinary time series. Commonly, stationary data is more conducive to prediction for time series. Therefore, we acquire a stationary time series S' by differencing the original time series, and then use the feature extractor H to transform S' into the feature vector $\mathbf{f}'$. Finally, $\tilde{\mathbf{f}}$ and $\mathbf{f}'$ are concatenated to obtain the concatenated vector $\mathbf{f}^{\mathbf{f}}$, and with $\mathbf{f}^{\mathbf{f}}$, the final prediction is carried out. Note that, the previous description shows the feature extractor H has two roles, one is to transform $\bar{S}$ to $\mathbf{f}$ and the other is to convert S' to $\mathbf{f}'$.

According to the aforementioned idea, we propose the Pre-SMATS whose framework is shown in Fig. 5. The Pre-SMATS consists of three components: *Feature Extractor*, *Sub-task Classifier* and *Main Task Predictor*. The feature extractor is responsible for transforming the data $\bar{S}$ and the stationary time series S' into the corresponding feature vectors $\mathbf{f}$ and $\mathbf{f}'$. The sub-task classifier has two functions, one is to perform the data stage classification task, i.e., to predict the stage of the data at the next moment, and the other is to output the stage vector $\tilde{\mathbf{f}}$ learned in the classification process into the main task predictor, which is responsible for the final prediction, concatenates the feature vector $\mathbf{f}'$ with the stage vector $\tilde{\mathbf{f}}$ output by the sub-task to form the concatenated vector $\mathbf{f}^{\mathbf{f}}$, with which to complete the prediction task. The learning block of Pre-SMATS can be fully connected layers, recurrent and convolutional layers, or other neural network layers, and the most appropriate one can be selected according to the task requirements and data features. In the remainder of this section, each component of the model is described in detail and shown how they work together to accomplish the final prediction task.

3.2. Feature extractor

In our experiments, we chose the Bi-LSTM as the learning block for the feature extractor, and for a fair comparison with baselines, no special improvements are made to the learning block. Fig. 6 provides an overview of the feature extractor, whose two branches are both composed of Bi-LSTM. The upper branch is to extract the features of the series $\bar{S}$, the bottom one is to extract the feature of the stationary time series S'. $\bar{S}$ (S') is input to the Bi-LSTM to obtain $\mathbf{f}$ ($\mathbf{f}'$), which is the feature vector of input.

3.3. Sub-task classifier

From the sub-task classifier structure indicated in Fig. 7, whose learning block consists of the Bi-LSTM layer, the FC layer, and the GlobalMaxpooling1D layer, we can find that the feature vector $\mathbf{f}$ generated by the feature extractor is input and transmitted to the Bi-LSTM layer, and then the stage vector $\tilde{\mathbf{f}}$ which is the embedding of stage characteristics is generated, and $\tilde{\mathbf{f}}$ is input into the main task for auxiliary prediction.

Meanwhile, $\tilde{\mathbf{f}}$ is also input into the FC layer and the main role of the GlobalMaxpooling1D layer is to reduce the dimensionality so that unify dimensions with the output. After the GlobalMaxpooling1D layer, the yielded $\mathbf{t}^{\mathbf{p}}$ is passed to another FC layer for predicting the next stage, obtaining Y_c. The whole calculating process can be expressed as follows

$$\mathbf{Q_p} = \boldsymbol{\omega}_t \mathbf{t}^{\mathbf{p}} + \mathbf{b} \tag{7}$$

$$Y_c(\mathbf{f}) = Softmax\left(\mathbf{Q_p}\right) \tag{8}$$

where $\boldsymbol{\omega}_t$ represents the learned parameters, $\mathbf{b}$ is the bias, and the activation function is set to $Softmax$ to obtain the probability of each stage Y_c.

We use the softmax cross-entropy loss function to calculate the sub-task loss as shown below:

$$L_p\left(h, Y_c\right) = -\frac{1}{n} \sum_{i=1}^{n} \sum_{j=1}^{m} h_{ij} \cdot \log Y_{c_{ij}} \tag{9}$$

where n is the total number of samples in the training set, m is the number of stages, h_{ij} is the label value of the jth stage of the ith sample, and $Y_{c_{ij}}$ is the predicted value of h_{ij}.

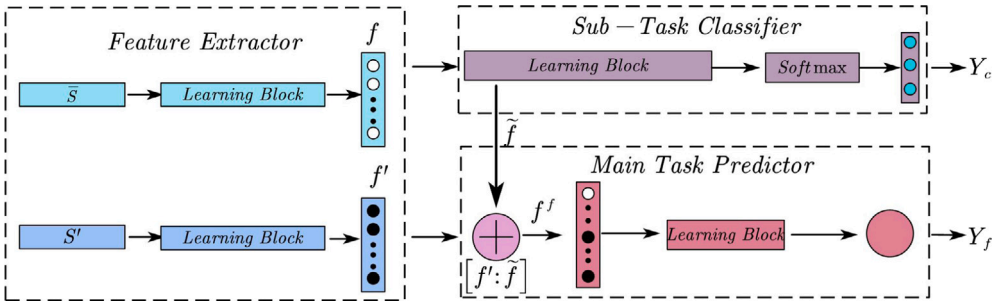


Fig. 5. The framework of Pre-SMATS model.

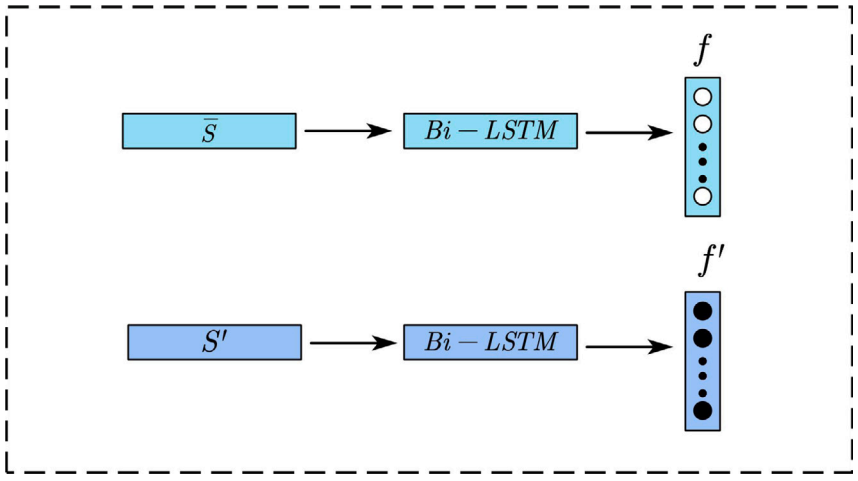


Fig. 6. The structure of Feature Extractor.

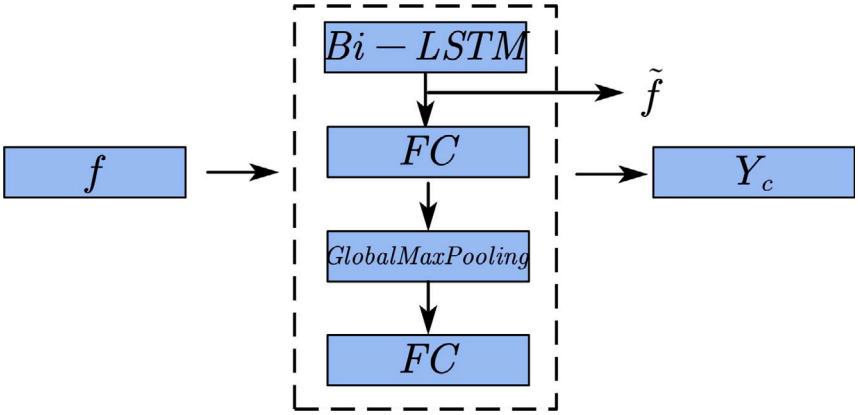


Fig. 7. The structure of Sub-task Classifier.

3.4. Main task predictor

The previous description shows the stage vector, the embedding of stage characteristics, have been obtained in the sub-task classifier and entered into the main task predictor. In order to improve the accuracy of the final prediction, we also need to calculate the feature vector of the time series itself. After being differenced, the original time series S is transformed into stationary time series S'. Thus, we employ S' to better extract the feature of the data, which is completed by the feature extractor. Namely, the feature vector $\mathbf{f}'$ is computed after S' is fed into the feature extractor, see the bottom branch shown in Fig. 6.

Fig. 8, which presents the structure of the main task predictor, indicates that the predictor starts with the concatenation of $\mathbf{f}'$ and $\tilde{\mathbf{f}}$ which generated during the process of the sub-task, to obtain the concatenated vector $\mathbf{f}^{\mathbf{f}}$, i.e. $\mathbf{f}^{\mathbf{f}} = [\mathbf{f}' : \tilde{\mathbf{f}}]$. Then, $\mathbf{f}^{\mathbf{f}}$ is passed to the FC layer and the GlobalMaxpooling1D layer (the learning block of main task predictor) for learning, and the prediction result Y_f is returned by the final FC layer.

In model training, the Huber loss function is leveraged to calculate the loss. Compared to ordinary least squares, Huber loss reduces the penalty on outliers and has the advantages of both MAE and MSE. It also overcomes the disadvantages of both MAE and MSE, that is, it not only maintains a continuous derivative of the loss function, but also acquires more accurate minimum value and better robustness to

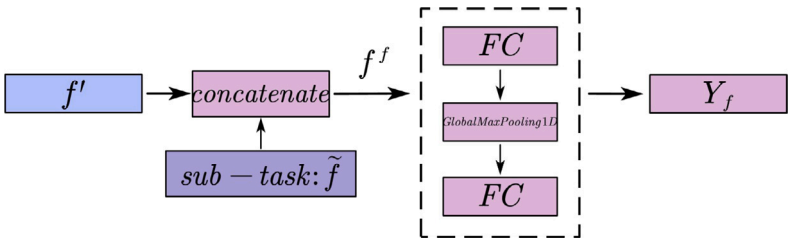


Fig. 8. The structure of Main Task Predictor.

outliers. The equation is specifically expressed as:

$$L_d\left(r, Y_f\right) = \frac{1}{n}\sum_{i=1}^{n}\begin{cases} \frac{1}{2}\left(r_i - Y_{f_i}\right)^2 & \left|r_i - Y_{f_i}\right| \le \delta \\ \delta\left|r_i - Y_{f_i}\right| - \frac{1}{2}\delta^2 & otherwise \end{cases} \tag{10}$$

where n is the total number of samples in the training set, r_i is the true value of ith sample, and Y_{f_i} is the predicted value of r_i. Combining Eq. (9) with Eq. (10), we can formulate the loss function for the whole model as:

$$L = w_d L_d + w_p L_p \tag{11}$$

where w_d, w_p are the weights of L_d and L_p, respectively.

Note that, in the process of multi-task training, we used the minimized L as the selection criterion of the optimal model. That is, the parameters that the model finally saves are:

$$\left(\hat{\omega}, \hat{\theta}\right) = arg \min_{\omega,\theta} L\left(\omega, \theta\right) \tag{12}$$

where ω, θ are the parameters generated during the training of the model.

4. Experiments

To better evaluate the model, we compared our proposed Pre-SMATS model with six representative existing seasonal time series prediction models, SARIMA, TCN, LSTM, Bi-LSTM, LSTM+Attention and Bi-LSTM+Attention which were discussed in Section 2 by using the Chinese civil aviation passenger traffic (CCAPT) dataset, a small multi-stage seasonal time series. To further verify the generalization capability of model, we also conducted experiments on the dataset of IC board production furnace temperature curve (FTC), a small multi-stage time series, which has only stage characteristics but not seasonal characteristics. This group of experiments take ARIMA, TCN, LSTM, Bi-LSTM, LSTM+Attention and Bi-LSTM+Attention as baseline models to explore the accuracy of the Pre-SMATS model on small multi-stage time series.

4.1. Datasets

- *CCAPT: the dataset of Chinese civil aviation passenger traffic* This dataset was collected from the official website of the Ministry of Transport of China[1] and records the domestic passenger traffic of China's civil aviation of each month from 2013–2019. The dataset is a small multi-stage seasonal time series with a period of 12 months that can be divided into 3 stages, January–May, June–August and September–December. Table 1 gives the detailed information of the dataset.
- *FTC: the dataset of furnace temperature curve* This dataset is the value of the center temperature of the welding zone generated in an actual production process of an integrated circuit board,[2] which is collected by the sensors on the product moving over time. We can use a curve to describe the corresponding relationship between furnace temperature and time, this curve is called the furnace temperature curve, that is, the center of the welding zone temperature curve. It can be divided into five stages represented by 1, 2, 3, 4, and 5, respectively. The specific information of the dataset is described in Table 2.

Table 1
Detailed information of CCAPT dataset.

Date	Number	Stage
2013–1	2376.7	1
2013–2	2582	1
…	…	…
2013–5	2651	1
2013–6	2650	2
…	…	…
2013–8	3204.3	2
2013–9	2806.6	3
…	…	…
2013–12	2567.4	3
2014–1	2828.9	1
2014–2	2876.5	1
…	…	…
2019–10	5093.8	3
2019–11	4715.4	3
2019–12	4643.6	3

Table 2
Detailed information of FTC dataset.

Seconds	Temperature	Stage
19.5	30.48	1
20	30.95	1
…	…	…
48.5	81.37	1
49	82.28	2
…	…	…
78.5	123.58	2
79	124.09	3
…	…	…
213.5	189.4	3
214	190.02	4
…	…	…
338.5	192.89	4
339	192.08	5
…	…	…
373	143.79	5

4.2. Data pre-processing

Before inputting the data into the model, some pre-processing need to be carried on the data, including stage division, data differencing, data standardization and using sliding windows to load the data.

Stage division: In order to obtain the embedding of stage characteristics, we first divide the time series in the dataset into stages. After investigation, we found that the CCAPT dataset has seasonal

[1] http://www.mot.gov.cn/tongjishuju/minhang/index.html
[2] http://www.mcm.edu.cn/html_cn/node/10405905647c52abfd6377c0311632b5.html

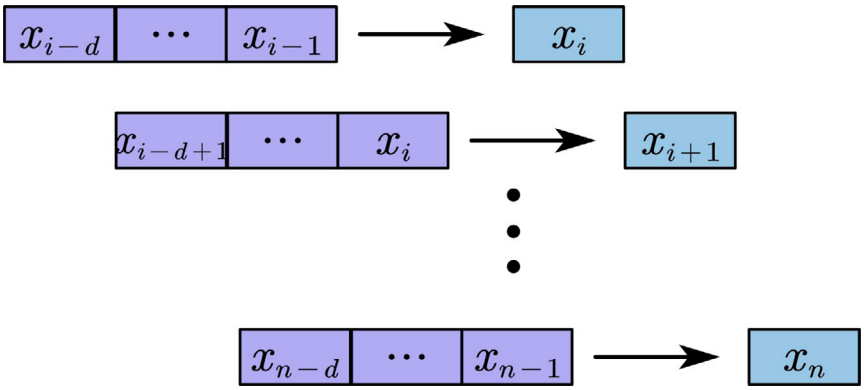


Fig. 9. Predicting data with sliding window.

characteristics with upward growth trend, and therefore, we performed first-order differencing on it using Eq. (13) with $z = 1$ to get the incremental series and conducted stage division on the incremental series. The FTC data is a general time series and can be divided into stages directly on the original data. According to Definition 2, when dividing the incremental series of CCAPT, we set $w = 3$, $k = 1$, $\beta = 0.9$, and for dividing the FTC dataset, we set $w = 60$, $k = 10$ and $\beta = 0.005$, and some of the results are shown in Table 1 and Table 2, respectively.

Data differencing: According to the previous description of the main task predictor (see Section 3.4), we know that in order to accurately complete the final prediction, in addition to input the stage vector generated by the sub-task to the main task, the feature vector extracted from the stationary time series S' should also be entered. We use the following formula to difference the original data

$$s'_{t_i} = s_{t_i} - s_{t_{i-z}} \tag{13}$$

where s'_{t_i} is the result after differencing of s_{t_i}, s_{t_i} is the value of the original time series S at time point t_i, and z is the difference step. In CCAPT dataset, z is 12 to obtain S', and in FTC dataset, z is 1 to obtain S'.

Data standardization: In order to speed up the training of the model and improve the accuracy of calculation, it is necessary to standardize the data before training and turn it into a distribution with a mean value of 0 and a standard deviation of 1. The formula is as follows

$$x' = \frac{x - mean(\hat{x})}{\sigma} \tag{14}$$

where x is $s'_{t_i} \in S'$, x' denotes the standardized data, $\hat{x}$ represents the training set that is divided from S', *mean* is the mean function, and σ is the standard deviation of the training set. It is important to note that, we do not standardization the input data $\bar{S}$ in order to preserve the original stage characteristics.

Sliding window: Sliding window is a general approach for loading history data to predict. Fig. 9 demonstrates the process of predicting the next data using a sliding window with size of d. In CCAPT, d is 3, and in FTC, d is 6.

4.3. Experimental setup

On the CCAPT dataset, for the feature extractor shown in Fig. 6, the LSTM output dimension of the upper branch is set to 16, and the LSTM output of the lower branch is set to 8, so the output of Bi-LSTM has 32 and 16 dimensions, respectively. In the sub-task classifier shown in Fig. 7, the output vector $\tilde{\mathbf{f}}$ of Bi-LSTM is 32 dimensions and the output of the FC layer has 16 dimensions. The FC layer with a $Softmax$ activation function is used to perform the stage classification, and in order to determine the probability of data belonging to different stages, the number of its output dimensions need to be same as the number of stages, that is, it need to be set to 3. In the main task predictor depicted in Fig. 8, after concatenating the feature vector $\mathbf{f}'$ output from the feature extractor with the stage vector $\tilde{\mathbf{f}}$ generated in the sub-task, the obtained 48-dimensional vector $\mathbf{f}^{\mathbf{f}}$ is passed to the FC layer with *tanh* activation function and output a 16-dimensional vector, which is input to GlobalMaxPooling1D layer to reduce the dimensions, then the last FC layer calculates the final prediction result Y_f. The w_d and w_p of loss functions, Eq. (11), are separately set to 5 and 1.

On the FTC dataset, for the feature extractor, the output dimension of LSTM for both branches is set to 16, and therefore the output of Bi-LSTM become 32-dimensional. In the sub-task classifier, the output vector $\tilde{\mathbf{f}}$ of Bi-LSTM is 32 dimensions and the output vector of the FC layer has 16 dimensions. For the FC layer with a $Softmax$ activation function, the number of its output dimensions is set to 5. In the main task predictor, after concatenating the feature vector $\mathbf{f}'$ output from the feature extractor with the stage vector $\tilde{\mathbf{f}}$ generated in the sub-task, the obtained 64-dimensional vector $\mathbf{f}^{\mathbf{f}}$ is passed to the FC layer with *tanh* activation function to learn and output 32-dimensional vector, which is put into GlobalMaxPooling1D layer to reduce the dimensions, then the last FC layer calculates the final prediction result Y_f. The values of w_d and w_p are both 1.

The optimizer used in the experiments is Adam, which has a learning rate of 1e-2, epsilon 1e-5, and the batch size is set to 32. The maximum epoch for CCAPT is 50 and for FTC is 100. All experiments were conducted on the Cloud computing platform with a GPU:1*V100 (32G), CPU:8 cores 64G. All the programs are implemented in Python 3.7 in the environment of TensorFlow-2.0.0.

4.4. Evaluation indicators

The metrics we employed to evaluate the performance of models are MSE (mean squared error), RMSE (root mean squared error), MAE (mean absolute error), MAPE (mean absolute percentage error) and MSLE (mean squared log error), which are formulated as follows

$$MSE(r, Y_f) = \frac{1}{n}\sum_{i=1}^{n}\left(r_i - Y_{f_i}\right)^2 \tag{15}$$

$$RMSE(r, Y_f) = \sqrt{\frac{1}{n}\sum_{i=1}^{n}\left(r_i - Y_{f_i}\right)^2} \tag{16}$$

$$MAE(r, Y_f) = \frac{1}{n}\sum_{i=1}^{n}\left|\left(r_i - Y_{f_i}\right)\right| \tag{17}$$

$$MAPE(r, Y_f) = \frac{100\%}{n}\sum_{i=1}^{n}\left|\frac{r_i - Y_{f_i}}{r_i}\right| \tag{18}$$

$$MSLE(r, Y_f) = \frac{1}{n}\sum_{i=1}^{n}\left(\log\left(r_i + 1\right) - \log\left(Y_{f_i} + 1\right)\right)^2 \tag{19}$$

where n is the number of samples, r_i is the real value of the ith sample, and Y_{f_i} is the predicted value of r_i.

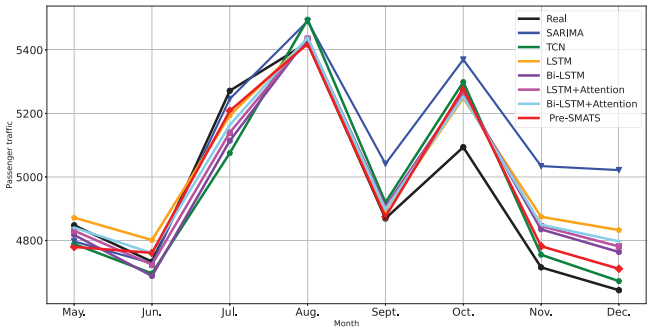


Fig. 10. Prediction results of different models on CCAPT dataset.

4.5. Experimental results

On CCAPT dataset, we test the performance of the comparative models for predicting the passenger traffic of the next eight months. Fig. 10 shows the predicted results of each model, and Table 3 compares the performance of each model. On the dataset of FTC, in the experiment, to verify the effectiveness of the models, we used the 10% of the data as the test set, and Fig. 11 shows the predicted results of each model. To highlight the difference of prediction results of comparative models on this dataset, part of the prediction results are compared in Fig. 11(a). Table 4 lists the performance comparison of various models. The experimental results show that the accuracy of Pre-SMATS model is better than that of the baseline models on both datasets.

On the CCAPT dataset, i.e. a small multi-stage seasonal time series, the Pre-SMATS model is significantly superior to the baseline models on the prediction accuracy. As can be seen in Fig. 10, although Pre-SMATS does not perform best in all months, its forecasts are the most stable and do not produce forecasts that deviate significantly from actual values. According to the data in Table 3, it can be calculated that from LSTM to Bi-LSTM + Attention, MSE decreases by 17%, RMSE by 9%, MAE by 10%, MAPE by 11% and MSLE by 18%. Compared with Bi-LSTM +Attention, the errors of Pre-SMATS are reduced by 34%, 19%, 24%, 23% and 39%, respectively. The above analysis illustrates that Pre-SMATS has significant advantages in predicting small multi-stage seasonal time series, and the improvement in input data is more remarkable than the effect brought by model optimization.

Similarly, on the FTC dataset, a general small multi-stage time series, the Pre-SMATS model outperforms the baseline models, with prediction curves closer to the real data than the comparison models, as shown in Fig. 11(b). Table 4 indicates that the best-performing baseline model on FTC is TCN, compared with it, the error metrics MSE, RMSE, MAE, MAPE and MSLE of Pre-SMATS decreased by 42%, 25%, 16%, 18%, and 43%, respectively. Although the prediction effect of different baseline models has been significantly improved and good results have been achieved, Pre-SMATS may still be optimized to further reduce the prediction error.

In Section 2.1, we have discussed the drawback of ARIMA that it is not suitable for long-term forecasting. For a time series that produces relatively dense data in a short period of time, it leads to a significant decrease in the accuracy of the later forecasts, which is fatal to the accuracy of the forecasts, as also verified in the experimental results (see Fig. 11(a)).

The aforementioned analysis shows that, compared with the baseline models, Pre-SMATS has a better performance in the prediction tasks on small multi-stage seasonal time series as well as small general multi-stage time series, which demonstrates that it has a good generalization capability.

Table 3
Performance comparison of the Pre-SMATS model with the baseline models on the CCAPT dataset.

Models	MSE	RMSE	MAE	MAPE	MSLE
SARIMA	44755.210	211.554	161.693	3.341%	0.00180
TCN	11942.58	109.282	86.018	1.696%	0.00044
LSTM	12139.201	110.178	88.707	1.828%	0.00050
Bi-LSTM	10902.528	104.415	86.773	1.756%	0.00043
LSTM+Attention	10518.686	102.560	80.237	1.628%	0.00042
Bi-LSTM+Attention	10075.793	100.378	79.286	1.617%	0.00041
Pre-SMATS	**6578.155**	**81.105**	**60.850**	**1.232%**	**0.00025**

Table 4
Performance Comparison of the Pre-SMATS model with the baseline models on the FTC dataset.

Models	MSE	RMSE	MAE	MAPE	MSLE
ARIMA	185.45816	13.6183	8.4530	3.6032%	0.00305
TCN	7.10e−05	0.0084	0.0065	0.0027%	1.24e−09
LSTM	0.0034	0.0588	0.0456	0.0191%	6.04e−08
Bi-LSTM	0.0022	0.0478	0.0393	0.0164%	3.95e−08
LSTM+Attention	0.00035	0.0187	0.0154	0.0064%	6.13e−09
Bi-LSTM+Attention	0.00023	0.0152	0.0124	0.0052%	4.09e−09
Pre-SMATS	**4.06e−05**	**0.0063**	**0.0054**	**0.0022%**	**7.00e−10**

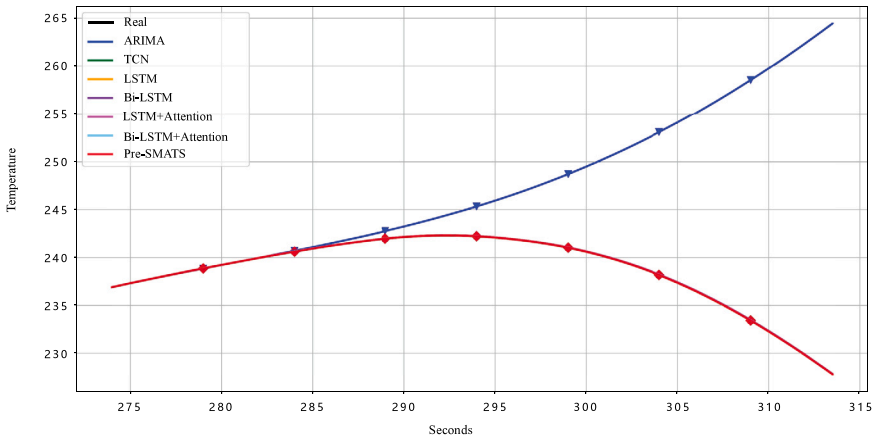
Table 5
Performance comparison of different architectures for Pre-SMATS on CCAPT dataset.

Models	MSE	RMSE	MAE	MAPE	MSLE
With single input	12005.818	109.571	85.702	1.757%	0.00049
Without sub-task classifier	7723.378	87.882	75.069	1.511%	0.00031
Pre-SMATS	**6578.155**	**81.105**	**60.850**	**1.232%**	**0.00025**

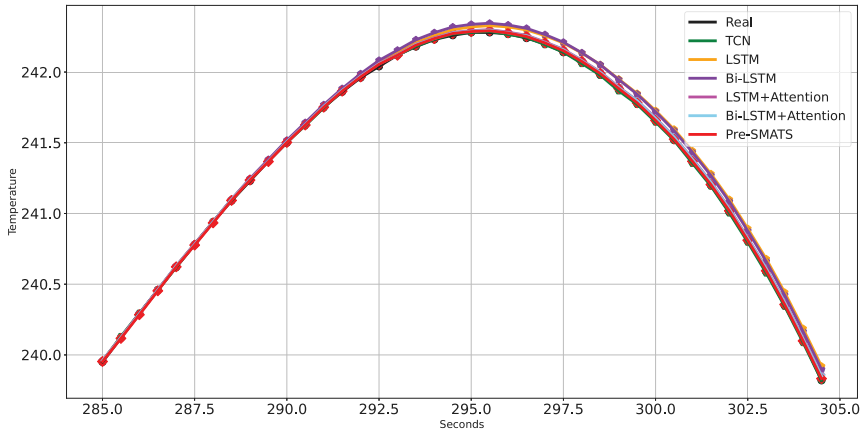
4.6. Ablation analysis

In this section, we focus on exploring whether the sub-task classifier added to the Pre-SMATS model displayed in Fig. 6 on the basis of general prediction model has a positive effect on prediction. In the experiment, we set up two control groups, one is the model *With single input*, which only has the bottom branch with the stationary time series S' as input and does not include the upper branch of feature extractor and sub-task classifier, and the other is the model *Without sub-task classifier* which removes the sub-task classifier and only conducts the concatenation of $\mathbf{f}$ and $\mathbf{f}'$, i.e. only concatenating the feature vectors of the two inputs without taking advantages of the relationship between the incremental series $\bar{S}$ and its stages, i.e. the stage vector, to detect if the stage vector work.

Compared the experimental results of *With single input* shown in Table 5 with that of the existing prediction models shown Table 3, adding FC layer on the basis of Bi-LSTM makes almost no optimization on the prediction, which indicates that the improvement on prediction performance is the role of other components of the model. In contrast, Pre-SMATS reduces MSE, RMSE, MAE, MAPE and MSLE by 45%, 25%, 29%, 29% and 48%, which proves that the introduction of sub-task classifier and $\bar{S}$ contribute a lot to the improvement of predication. On the other hand, Table 5 illustrates, compared *Without sub-task classifier* with *With single input*, by increasing $\bar{S}$ as input, the predicting effect is improved by 35%, 20%, 11%, 14%, and 36% on the five metrics, which suggests that the combination of the stationary time series S' and incremental time series $\bar{S}$ makes help to predict the seasonal time series. Further, Pre-SMATS is superior to *Without sub-task classifier* on the five metrics by 14%, 6%, 20%, 18%, and 19%, respectively, showing that the addition of sub-task significantly improves the accuracy of prediction, which means using the information of the incremental series $\bar{S}$ with that of its stages can make positive contribute to the prediction results.



(a) Overall prediction results for all models



(b) Partial prediction results of each model except ARIMA

Fig. 11. Prediction results of different models on FTC dataset.

5. Conclusion and future works

Due to the insufficiency of data and the acquisition of seasonal characteristics, it is difficult for existing prediction models to achieve ideal prediction effect for small seasonal time series. In this paper, the implicit stage information on the original time series is obtained to enhance data features, so as to alleviate the negative impact of insufficient samples on training effect. To improve the accuracy of prediction, the seasonal characteristics of time series can be expressed in a finer granularity by using the information of stages in series. Combined with multi-task learning, Pre-SMATS model is proposed, which consists of feature extractor, sub-task classifier and main task predictor. The model optimizes the main task predictor by using the stage vector obtained from sub-task. Compared with the best-performing baseline neural network model, by capturing the information of stages, the five error metrics MSE, RMSE, MAE, MAPE, and MSLE of Pre-SMATS reduced 34%, 19%, 24%, 23% and 39% on the CCAPT dataset, a small multi-stage seasonal time series, and 42%, 25%, 16%, 18% and 43% on the FTC dataset, a general small multi-stage time series, demonstrates that our Pre-SMATS has good performance and generalization ability in prediction small multi-stage time series. However, as we previously mentioned, the selection of appropriate learning blocks is important to the improvement of the performance of our proposed model, but how to find it efficiently is an issue worthy of study, and we will conduct in-depth research on this problem in the future work.

CRediT authorship contribution statement

Shiling Wu: Writing – original draft, Writing – editing, Coding, Validation. **Dunlu Peng:** Conceptualization, Formal analysis, Funding acquisition, Writing - review.

Declaration of competing interest

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

Acknowledgments

The work is supported by the National Natural Science Foundation of China under Grant No. 61772342. We would like to express our special thanks to the members in our lab for their valuable discussion on this work.

References

Abuqaddom, I., Mahafzah, B. A., & Faris, H. (2021). Oriented stochastic loss descent algorithm to train very deep multi-layer neural networks without vanishing gradients. *Knowledge-Based Systems*, *230*, Article 107391. http://dx.doi.org/10.1016/j.knosys.2021.107391.

Alom, M. Z., Aspiras, T., Taha, T. M., Bowen, T., & Asari, V. K. (2020). MitosisNet: End-to-end mitotic cell detection by multi-task learning. *IEEE Access*, *8*, 68695–68710. http://dx.doi.org/10.1109/ACCESS.2020.2983995.

Ashour, M. A. H., & Abbas, R. A. (2018). Improving time series' forecast errors by using recurrent neural networks. In *Proceedings of the 2018 7th international conference on software and computer applications* (pp. 229–232). http://dx.doi.org/10.1145/3185089.3185151.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473. URL https://arxiv.53yu.com/abs/1409.0473.

Bai, S., Kolter, J. Z., & Koltun, V. (2018). An empirical evaluation of generic convolutional and recurrent networks for sequence modeling. CoRR, arXiv:1803.01271.

Box, G. E., Jenkins, G. M., & Reinsel, G. (1970). Time series analysis: forecasting and control holden-day san francisco. BoxTime Series Analysis: Forecasting and Control Holden Day1970.

Carmona-Benítez, R. B., & Nieto, M. R. (2020). SARIMA damp trend grey forecasting model for airline industry. *Journal of Air Transport Management*, *82*, Article 101736. http://dx.doi.org/10.1016/j.jairtraman.2019.101736.

Caruana, R. (1997). Multitask learning. *Machine Learning*, *28*(1), 41–75. http://dx.doi.org/10.1023/A:1007379606734.

Du, B., Zhou, Q., Guo, J., Guo, S., & Wang, L. (2021). Deep learning with long short-term memory neural networks combining wavelet transform and principal component analysis for daily urban water demand forecasting. *Expert Systems with Applications*, Article 114571. http://dx.doi.org/10.1016/j.eswa.2021.114571.

du Jardin, P. (2021). Forecasting corporate failure using ensemble of self-organizing neural networks. *European Journal of Operational Research*, *288*(3), 869–885. http://dx.doi.org/10.1016/j.ejor.2020.06.020.

El-Sappagh, S., Abuhmed, T., Islam, S. R., & Kwak, K. S. (2020). Multimodal multitask deep learning model for alzheimer's disease progression detection based on time series data. *Neurocomputing*, *412*, 197–215. http://dx.doi.org/10.1016/j.neucom.2020.05.087.

Fan, D., Sun, H., Yao, J., Zhang, K., Yan, X., & Sun, Z. (2020). Well production forecasting based on ARIMA-LSTM model considering manual operations. *Energy*, Article 119708. http://dx.doi.org/10.1016/j.energy.2020.119708.

Fang, W., Zhang, S., Huang, H., Dang, S., Huang, Z., Li, W., et al. (2020). Learn to make decision with small data for autonomous driving: Deep Gaussian process and feedback control. *Journal of Advanced Transportation*, *2020*, http://dx.doi.org/10.1155/2020/8495264.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, *33*(4), 917–963. http://dx.doi.org/10.1007/s10618-019-00619-1.

Fawaz, H. I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., et al. (2020). Inceptiontime: Finding alexnet for time series classification. *Data Mining and Knowledge Discovery*, *34*(6), 1936–1962. http://dx.doi.org/10.1007/s10618-020-00710-y.

Gautam, Y. (2021). Transfer learning for COVID-19 cases and deaths forecast using LSTM network. *ISA Transactions*, http://dx.doi.org/10.1016/j.isatra.2020.12.057.

Hayashi, S., Tanimoto, A., & Kashima, H. (2019). Long-term prediction of small time-series data using generalized distillation. In *2019 International joint conference on neural networks* (pp. 1–8). IEEE, http://dx.doi.org/10.1109/IJCNN.2019.8851687.

Hewage, P., Behera, A., Trovati, M., Pereira, E., Ghahremani, M., Palmieri, F., et al. (2020). Temporal convolutional neural (TCN) network for an effective weather forecasting using time-series data from the local weather station. *Soft Computing*, *24*(21), 16453–16482. http://dx.doi.org/10.1007/s00500-020-04954-0.

Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*, *9*(8), 1735–1780. http://dx.doi.org/10.1162/neco.1997.9.8.1735.

Hua, Y., Zhao, Z., Li, R., Chen, X., Liu, Z., & Zhang, H. (2019). Deep learning with long short-term memory for time series prediction. *IEEE Communications Magazine*, *57*(6), 114–119. http://dx.doi.org/10.1109/MCOM.2019.1800155.

Hyndman, R. J. (2011). Cyclic and seasonal time series. Hyndsight Blog, URL https://robjhyndman.com/hyndsight/cyclicts/.

Ismail, E. A. A. (2019). Forecasting the number of arab and foreign tourists in Egypt using ARIMA models. *International Journal of Systems Assurance Engineering and Management*, 1–5. http://dx.doi.org/10.1007/s13198-019-00873-y.

Istaiteh, O., Owais, T., Al-Madi, N., & Abu-Soud, S. (2020). Machine learning approaches for COVID-19 forecasting. In *2020 International conference on intelligent data science technologies and applications* (pp. 50–57). IEEE, http://dx.doi.org/10.1109/IDSTA50958.2020.9264101.

Jin, X., Yu, X., Wang, X., Bai, Y., Su, T., & Kong, J. (2020). Prediction for time series with CNN and LSTM. In *Proceedings of the 11th international conference on modelling, identification and control* (pp. 631–641). Springer, http://dx.doi.org/10.1007/978-981-15-0474-7_59.

Keshari, R., Ghosh, S., Chhabra, S., Vatsa, M., & Singh, R. (2020). Unravelling small sample size problems in the deep learning world. In *2020 IEEE sixth international conference on multimedia big data* (pp. 134–143). IEEE, URL https://arxiv.53yu.com/pdf/2008.03522.pdf.

Kulshrestha, A., Krishnaswamy, V., & Sharma, M. (2020). BayesIan BILSTM approach for tourism demand forecasting. *Annals of Tourism Research*, *83*, Article 102925. http://dx.doi.org/10.1016/j.annals.2020.102925.

Lea, C., Flynn, M. D., Vidal, R., Reiter, A., & Hager, G. D. (2017). Temporal convolutional networks for action segmentation and detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 156–165). URL https://openaccess.thecvf.com/content_cvpr_2017/papers/Lea_Temporal_Convolutional_Networks_CVPR_2017_paper.pdf.

Li, L., Zhang, J., Wang, Y., & Ran, B. (2018). Missing value imputation for traffic-related time series data based on a multi-view learning method. *IEEE Transactions on Intelligent Transportation Systems*, *20*(8), 2933–2943. http://dx.doi.org/10.1109/TITS.2018.2869768.

Liang, Y., Ke, S., Zhang, J., Yi, X., & Zheng, Y. (2018). Geoman: Multi-level attention networks for geo-sensory time series prediction. In *IJCAI* (pp. 3428–3434). URL http://dx.doi.org/10.24963/ijcai.2018/476.

Lim, S., Kim, S. J., Park, Y., & Kwon, N. (2021). A deep learning-based time series model with missing value handling techniques to predict various types of liquid cargo traffic. *Expert Systems with Applications*, *184*, Article 115532. http://dx.doi.org/10.1016/j.eswa.2021.115532.

Lima, S., Gonçalves, A. M., & Costa, M. (2019). Time series forecasting using holt-winters exponential smoothing: An application to economic data. *AIP Conference Proceedings*, *2186*(1), Article 090003. http://dx.doi.org/10.1063/1.5137999.

Lin, L., Xu, B., Wu, W., Richardson, T. W., & Bernal, E. A. (2019). Medical time series classification with hierarchical attention-based temporal convolutional networks: A case study of myotonic dystrophy diagnosis. In *CVPR workshops* (pp. 83–86). URL https://openaccess.thecvf.com/content_CVPRW_2019/papers/Explainable%20AI/Lin_Medical_Time_Series_Classification_with_Hierarchical_Attention-based_Temporal_Convolutional_Networks_CVPRW_2019_paper.pdf.

Livieris, I. E., Pintelas, E., & Pintelas, P. (2020). A CNN–LSTM model for gold price time-series forecasting. *Neural Computing and Applications*, *32*(23), 17351–17360.

Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2019). Neural network architecture based on gradient boosting for IoT traffic prediction. *Future Generation Computer Systems*, *100*, 656–673. http://dx.doi.org/10.1016/j.future.2019.05.060.

Lopez-Martin, M., Carro, B., & Sanchez-Esguevillas, A. (2020). IoT type-of-traffic forecasting method based on gradient boosting neural networks. *Future Generation Computer Systems*, *105*, 331–345. http://dx.doi.org/10.1016/j.future.2019.12.013.

Lopez-Martin, M., Sanchez-Esguevillas, A., Hernandez-Callejo, L., Arribas, J. I., & Carro, B. (2021). Additive ensemble neural network with constrained weighted quantile loss for probabilistic electric-load forecasting. *Sensors*, *21*(9), 2979. http://dx.doi.org/10.3390/s21092979.

Ma, T., & Tan, Y. (2020). Multiple stock time series jointly forecasting with multi-task learning. In *2020 International joint conference on neural networks* (pp. 1–8). IEEE, http://dx.doi.org/10.1109/IJCNN48605.2020.9207543.

Martínez, F., Frías, M. P., Pérez-Godoy, M. D., & Rivera, A. J. (2018). Dealing with seasonality by narrowing the training set in time series forecasting with kNN. *Expert Systems with Applications*, *103*, 38–48. http://dx.doi.org/10.1016/j.eswa.2018.03.005.

Mnih, V., Heess, N., Graves, A., & Kavukcuoglu, K. (2014). Recurrent models of visual attention. arXiv preprint arXiv:1406.6247. URL https://proceedings.neurips.cc/paper/2014/file/09c6c3783b4a70054da74f2538ed47c6-Paper.pdf.

Molina, M. A., Asencio-Cortés, G., Riquelme, J. C., & Martínez-Álvarez, F. (2020). A preliminary study on deep transfer learning applied to image classification for small datasets. In *International workshop on soft computing models in industrial and environmental applications* (pp. 741–750). Springer, http://dx.doi.org/10.1007/978-3-030-57802-2_71.

Montgomery, D. C., Johnson, L. A., & Gardiner, J. S. (1990). *Forecasting and time series analysis*. McGraw-Hill Companies.

Moreno-Barea, F. J., Jerez, J. M., & Franco, L. (2020). Improving classification accuracy using data augmentation on small data sets. *Expert Systems with Applications*, *161*, Article 113696. http://dx.doi.org/10.1016/j.eswa.2020.113696.

Noureen, S., Atique, S., Roy, V., & Bayne, S. (2019). Analysis and application of seasonal ARIMA model in energy demand forecasting: A case study of small scale agricultural load. In *2019 IEEE 62nd international midwest symposium on circuits and systems* (pp. 521–524). IEEE, http://dx.doi.org/10.1109/MWSCAS.2019.8885349.

Nwokike, C. C., Offorha, B. C., Obubu, M., Ugoala, C. B., & Ukomah, H. I. (2020). Comparing SANN and SARIMA for forecasting frequency of monthly rainfall in Umuahia. *Scientific African*, *10*, Article e00621. http://dx.doi.org/10.1016/j.sciaf.2020.e00621.

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. arXiv preprint arXiv:1704.02971, URL https://arxiv.53yu.com/abs/1704.02971.

Schuster, M., & Paliwal, K. K. (1997). Bidirectional recurrent neural networks. *IEEE Transactions on Signal Processing*, *45*(11), 2673–2681.

Selvaraj, J. J., Arunachalam, V., Coronado-Franco, K. V., Romero-Orjuela, L. V., & Ramírez-Yara, Y. N. (2020). Time-series modeling of fishery landings in the Colombian Pacific ocean using an ARIMA model. *Regional Studies in Marine Science*, *39*, Article 101477. http://dx.doi.org/10.1016/j.rsma.2020.101477.

Shankar, P., Arora, M. A., Kaushal, R., & Singh, I. (2019). Analyzing varied approaches for forecast of stock prices by combining news mining and time series analysis. In *2019 International conference on computing, power and communication technologies* (pp. 434–441). URL https://ieeexplore.ieee.org/abstract/document/8940491.

Song, H., Rajan, D., Thiagarajan, J., & Spanias, A. (2018). Attend and diagnose: Clinical time series analysis using attention models. In *Proceedings of the AAAI conference on artificial intelligence. Vol. 32. No. 1*. URL https://arxiv.org/pdf/1711.03905.pdf.

Tang, J., Liu, F., Zou, Y., Zhang, W., & Wang, Y. (2017). An improved fuzzy neural network for traffic speed prediction considering periodic characteristic. *IEEE Transactions on Intelligent Transportation Systems*, *18*(9), 2340–2350. http://dx.doi.org/10.1109/TITS.2016.2643005.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., et al. (2017). Attention is all you need. arXiv preprint arXiv:1706.03762. URL https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf.

Wei, W. W. (2006). Time series analysis. In *The Oxford handbook of quantitative methods in psychology. Vol. 2*. http://dx.doi.org/10.1093/oxfordhb/9780199934898.013.0022.

Wen, H., Zhang, J., Wang, Y., Lv, F., Bao, W., Lin, Q., et al. (2020). Entire space multi-task modeling via post-click behavior decomposition for conversion rate prediction. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval* (pp. 2377–2386). http://dx.doi.org/10.1145/3397271.3401443.

Yan, J., Mu, L., Wang, L., Ranjan, R., & Zomaya, A. Y. (2020). Temporal convolutional networks for the advance prediction of ENSO. *Scientific Reports*, *10*(1), 1–15. http://dx.doi.org/10.1038/s41598-020-65070-5.

Yan, H., & Ouyang, H. (2018). Financial time series prediction based on deep learning. *Wireless Personal Communications*, *102*(2), 683–700. http://dx.doi.org/10.1007/s11277-017-5086-2.

Yin, J., Deng, Z., Ines, A. V., Wu, J., & Rasu, E. (2020). Forecast of short-term daily reference evapotranspiration under limited meteorological variables using a hybrid bi-directional long short-term memory model (Bi-LSTM). *Agricultural Water Management*, *242*, Article 106386. http://dx.doi.org/10.1016/j.agwat.2020.106386.

Yin, W., Schütze, H., Xiang, B., & Zhou, B. (2016). Abcnn: Attention-based convolutional neural network for modeling sentence pairs. *Transactions of the Association for Computational Linguistics*, *4*, 259–272. http://dx.doi.org/10.1162/tacl_a_00097.

Yuan, H., & Li, G. (2021). A survey of traffic prediction: from spatio-temporal data to intelligent transportation. *Data Science and Engineering*, *6*(1), 63–85. http://dx.doi.org/10.1007/s41019-020-00151-z.

Zang, H., Liu, L., Sun, L., Cheng, L., Wei, Z., & Sun, G. (2020). Short-term global horizontal irradiance forecasting based on a hybrid CNN-LSTM model with spatiotemporal correlations. *Renewable Energy*, *160*, 26–41. http://dx.doi.org/10.1016/j.renene.2020.05.150.

Zeng, Y., Mao, H., Peng, D., & Yi, Z. (2019). Spectrogram based multi-task audio classification. *Multimedia Tools and Applications*, *78*(3), 3705–3722. http://dx.doi.org/10.1007/s11042-017-5539-3.

Zhang, L., Liu, P., Zhao, L., Wang, G., Zhang, W., & Liu, J. (2020). Air quality predictions with a semi-supervised bidirectional LSTM neural network. *Atmospheric Pollution Research*, http://dx.doi.org/10.1016/j.apr.2020.09.003.

Zhang, X., & Wang, J. (2018). A novel decomposition-ensemble model for forecasting short-term load-time series with multiple seasonal patterns. *Applied Soft Computing*, *65*, 478–494. http://dx.doi.org/10.1016/j.asoc.2018.01.017.

Zhang, Y.-w., Yuan, H.-w., Wu, H.-l., Sun, X., & Dong, Y.-c. (2020). Research on seasonal prediction of PM2. 5 based on PCA-BP neural network. *Journal of Physics: Conference Series*, *1486*(2), Article 022029. http://dx.doi.org/10.1088/1742-6596/1486/2/022029.

Zhang, B., Zhang, H., Zhao, G., & Lian, J. (2020). Constructing a PM2. 5 concentration prediction model by combining auto-encoder with Bi-LSTM neural networks. *Environmental Modelling & Software*, *124*, Article 104600. http://dx.doi.org/10.1016/j.envsoft.2019.104600.

Zhou, Y., Zhao, S., Wang, X., & Liu, W. (2018). Deep learning model and its application in big data. In *International conference of design, user experience, and usability* (pp. 795–806). Springer, http://dx.doi.org/10.1007/978-3-319-91797-9_55.