IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 26, 2019, accepted February 10, 2019, date of publication February 21, 2019, date of current version March 8, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2900371

An Ensemble Model Based on Adaptive Noise Reducer and Over-Fitting Prevention LSTM for Multivariate Time Series Forecasting

FAGUI LIU[1], MUQING CAI[1], LIANGMING WANG[2], AND YUNSHENG LU[1]
[1]School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China
[2]School of Software Engineering, South China University of Technology, Guangzhou 510006, China

Corresponding authors: Muqing Cai (cai.muqing@mail.scut.edu.cn) and Liangming Wang (lmwang@scut.edu.cn)

This work was supported in part by the Engineering and Technology Research Center of Guangdong Province for Logistics Supply Chain and Internet of Things under Project GDDST[2016]176, in part by the Key Laboratory of Cloud Computing for Super-integration Cloud Computing in Guangdong Province under Project 610245048129, and in part by the Engineering and Technology Research Center of Guangdong Province for Big Data Intelligent Processing under Project GDDST[2013]1513-1-11.

ABSTRACT Multivariate time series forecasting recently has received extensive attention with its wide application in finance, transportation, environment, and so on. However, few of the currently developed models have considered the impact of noise on prediction. Since multivariate time series contains multiple subsequences with strong nonlinear fluctuations, it is also difficult to obtain satisfactory prediction results. In this paper, aiming at improving prediction performance, we have proposed a novel ensemble three-phase model called adaptive noise reducer-stacked auto-encoder-validating-AdaBoost-based long short-term memory (ANR-SAE-VALSTM). We start with an introduction of a novel ANR for time series noise elimination. The SAEs are then used to extract features from the de-noised multivariate time series. Finally, we feed the de-noised features into the VALSTM to train an ensemble over-fitting prevention predictor. The proposed model is employed on the Beijing PM2.5 dataset and GEFCom2014 Electricity Price dataset. Compared with other popular models, the proposed model has achieved the best prediction performance in all prediction horizons. In addition, a careful ablation study is conducted to demonstrate the efficiency of our model design.

INDEX TERMS Multivariate time series forecasting, adaptive noise reducer, stacked auto-encoders, long short-term memory, validating AdaBoost algorithm.

I. INTRODUCTION

Multivariate time series forecasting has been widely applied in many fields, e.g., financial market forecasting [1]–[3], energy forecasting [4], [5], and environmental pollution forecasting [6]. It is of great significance to predict future new trends or potential hazardous events based on historical observations. For instance, accurate electricity price forecasting makes it possible to allow power generation plants to construct optimal price bidding strategy. Accurate environmental pollution forecasting can provide timely environmental quality information to assist in environmental management decisions. However, how to preprocess the complex chaotic multivariate time series, capture features among these variables, and make accurate predictions remain major challenges.

Multivariate time series is chaotic and noisy. For example, in the Beijing PM2.5 dataset, the target series is PM2.5, and the correlated variables include dew point (DEWP), temperature (TEMP), pressure (PRES), etc. Besides these measurable universal indicators, there are also noise factors such as local factory waste emissions and regional greening conditions related to PM2.5. However, these factors show randomness and abruptness, so it is difficult to consider them as related variables. Hassani *et al.* [7] pointed out that noise in time series seriously reduced the efficiency and effectiveness of analysis. Therefore, it is necessary to eliminate noise before analyzing the multivariate time series. However, due to the non-stationary and non-linear characteristics of the multivariate time series, traditional de-noising methods have

The associate editor coordinating the review of this manuscript and approving it for publication was Yongming Li.

many limits. Although the theory of frequency domain analysis is mature, it is still difficult to separate the time series of overlapping frequencies. The Wavelet Transform (WT) [8]–[10] is excellent in time-frequency localization. However, its de-noising ability depends on the selection of wavelet base function and the determination of threshold. Empirical Mode Decomposition (EMD) based de-noising method [11]–[13] applies EMD to decompose time series to multiple Intrinsic Mode Functions (IMFs) and a Residue (R). As the noise is concentrated in the high-frequency IMFs, only the noise reduction of the high-frequency IMFs is required. However, there are still two problems with EMD based de-noising method: (1) the difficulty to determine demarcation points of the high-frequency IMFs and low-frequency IMFs; and (2) how to de-noise the high-frequency IMFs. In order to address these two issues, we propose an Adaptive Noise Reducer (ANR) based on Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN). The idea of Permutation Entropy (PE) [14] is used to determine the demarcation point of high-frequency IMFs and low-frequency IMFs, and an adaptive soft threshold function is constructed to de-noise high-frequency IMFs.

Extracting the features of the time series and finding a more reasonable data input format help to reduce the training time and speed up the convergence [15], so it's essential to extract features before training predictor. Convolutional Neural Network (CNN) [16], [17] and Auto-Encoder (AE) [18], [19] are two effective time series feature extraction methods. CNN applies convolution layers on each variable time series aiming to extract distinctive features in the individual input time series. Although CNN adds a merged layer to concatenate the distinctive features, it still loses a lot of associated information. Furthermore, CNN is a kind of supervised learning. In the back propagation, the model needs to adjust both the weights of the predictor (e.g. MLP and RNN) and CNN, which leads to slowing convergence and falling into local optimum easily. AE is another effective method for time series feature extraction. AE is unsupervised, so feature extraction and time series prediction can be completely separated. Furthermore, AE treats the entire multivariate time series, which can fully take into account the relationship among multivariate time series so that the extracted features are more representative and robustness. However, single-layer AE can only construct simple function mapping, making it difficult to handle complex multivariate time series. In this paper, we use stacked auto-encoders (SAEs) [20] to extract features of multivariate time series layer by layer and fully consider the dependencies among multiple variables.

Time series analysis methods based on stochastic process theory and mathematical statistics have been difficult to deal with complex multivariate time series. In recent years, Recurrent Neural Networks (RNN) has become the most popular deep learning method for time series forecasting [21], [22]. The variant of RNN in particular, called Long Short-Term Memory (LSTM) [23] has significantly improved the prediction performance as it can effectively capture long- and short-term dependencies of time series. However, multivariate time series usually contains multiple subsequences with strongly nonlinear fluctuations, making it hard to obtain ideal prediction performance. To address this issue, many researches [24]–[26] apply the AdaBoost algorithm to combine a series of weak predictors to obtain a strong predictor, thus improving prediction accuracy. Nevertheless, these AdaBoost methods suffer from the over-fitting problem when used for time series forecasting. In each iteration of the AdaBoost algorithm, only the prediction error on the training set is considered so that the final weight of each predictor is also only related to the prediction error on the training set. As the weak predictor is continually added, the upper bound of the prediction error on the training set will continuously drop, and the risk of over-fitting will inevitably occur. To alleviate this issue, an improved Validating AdaBoost algorithm (V-AdaBoost) is proposed in this paper. V-AdaBoost algorithm adds the weight distribution of the validation set in each iteration. The weight of the predictor in each iteration is determined by both the prediction error on the training set and the validation set, which can largely prevent over-fitting and improve the generalization ability of the predictor.

The major contributions of this paper are summarized as follows:

1) A novel ensemble model called ANR-SAEs-VALSTM is proposed for multivariate time series forecasting. Its superiority and effectiveness are demonstrated by the experiments.
2) To tackle the problems of EMD based de-noising method mentioned above, an Adaptive Noise Reducer (ANR) is proposed. The idea of permutation entropy (PE) is used to determine the demarcation point among high-frequency IMFs and low-frequency IMFs, and an adaptive soft threshold function is constructed to de-noise the high-frequency IMFs.
3) For the reason that a more reasonable data input format helps to train the predictor more efficiently, SAEs is used to extract features, which fully considers the dependencies among multivariate time series.
4) To further enhance the forecasting performance and prevent over-fitting of LSTM model when using AdaBoost algorithm, Validating AdaBoost algorithm (V-AdaBoost) is proposed to combine LSTM predictors to obtain a strong predictor.

The remainder of this paper is organized as follows: Section II outlines the related work of time series forecasting models. Section III formulates the multivariate time series forecasting problem. The proposed model ANR-SAEs-VALSTM is introduced in section IV. The experiment and analysis are given in section V. Finally, we conclude our work and introduce the outlook for the future in section VI.

II. RELATED WORK

Time series forecasting models can be divided into two categories. The first is the traditional model based on stochastic process theory and mathematical statistics. The second is

based on neural networks. Traditional models usually use predefined linear or nonlinear models, and then dynamically adjust the parameters of the model based on the input data. The structure of neural network models is not fixed. They can flexibly explore the linear or nonlinear features based on the input data and learn the appropriate function mapping.

Autoregressive Moving Average Model (ARIMA) [27] is the most common traditional time series forecasting model. ARIMA is popular because of its prominent statistical properties and the well-known BOX-JenKins method. However, univariate ARIMA models are difficult for non-linear multivariate time series modeling. The Vector Autoregressive Model (VAR) [28] is therefore proposed to extend AR to multivariate time series modeling. However, VAR is still hard to deal with the dependencies among multivariate time series. For this reason, the elliptical VAR [29] model and the structured VAR [30] model are proposed to better resolve dependencies among multivariate time series. Nevertheless, the model capacity of VAR increases linearly over the temporal windows size, but the number of variables grows quadratically. A large number of variable parameters easily lead to over-fitting. Time series prediction problem can also be regarded as a standard variable parameter regression problem. It is therefore many machine learning methods with different loss function and regularization conditions are applied to time series forecasting. For example, Ridge regression with L2 regularization [31] is a technique for analyzing multiple regression data that suffers from multiple collinearities. It gets a more reliable and realistic result at the expense of abandoning the unbiasedness of the least squares method and losing part of the information. Support Vector Regression (SVR) [32] learns the max edge hyperplane based on the regression loss and controls the prediction error by the hyperparameter ε. However, SVR is global and not flexible enough to capture local trend features. Therefore, Local Support Vector Regression (LSVR) [33] is proposed to solve this problem by adjusting the margin locally and flexibly. Kernel Adaptive Filters (KAF) [34], [35] uses a linear adaptive filtering algorithm to achieve nonlinear signal processing and maps the input data to high-dimensional feature spaces. Besides parameter methods for modeling time series, there are also non-parametric methods which can reduce the computational complexity of the problem. For example, Gaussian Processes (GP) [36] is a non-parametric method and it models over a continuous domain of functions. The above traditional time series forecasting models use predefined models, and it is difficult to explore complex patterns in non-linear time series and dependencies among multiple variables. Therefore, in recent years, more and more works of literature analyze multivariate time series based on deep learning.

Neural network models, especially the Recurrent Neural Network (RNN), has been proved to be very suitable for modeling time series and capturing long- and short-term dependencies of nonlinear time series. In recent years, RNN has been widely used in time series forecasting. For example, Dasgupta and Osogami [37] proposed RNN-Gaussian DyBM which applies RNN to control the bias input of DyBM unit and derives a random gradient update rule. It enables the weights of the RNN to be trained online with other DyBM parameters. Qin *et al.* [38] proposed a Dual-Stage Attention-Based Recurrent Neural Network (DA-RNN). DA-RNN uses an input attention mechanism to adaptively extract relevant driving series according to the previous encoder hidden state. RNN based methods, however, easily suffer from the problem of gradient vanishing or gradient exploding when learning the long-term dependencies of time series [39]. Recently, Long Short-Term Memory (LSTM) [23] has achieved great success in many applications because its threshold structure can solve the gradient problem of RNN, such as the solar irradiance forecasting model (DWT-CNN-LSTM) [40], wind speed forecasting model (LSTMDE-HELM) [22], and stock index forecasting model (WSAEs-LSTM) [9]. Most of these models achieve better prediction accuracy by adding optimization methods to LSTM. In addition, Chang *et al.* [41] proposed a Memory Time-series Network (MTNet), which is a novel model based on LSTM. MTNet consists of a large memory component, three independent encoders, and an autoregressive component to train jointly. This model can capture long-term patterns of multivariate time series and is highly interpretable. Compared with traditional time series prediction models, neural network models have obvious advantages in capturing long- and short-term dependencies of multivariate time series, extracting features among them, and making accurate forecasting.

III. NOTATION AND PROBLEM STATEMENT

The multivariate time series is denoted as $X = \{X^{(1)}, X^{(2)}, \cdots, X^{(n)}\}$ where time series in X are correlated with each other. For example, in the Beijing PM2.5 dataset example, we have $X = \{X^{(1)}, X^{(2)}, \cdots, X^{(8)}\}$, where $X^{(1)}$ represents PM2.5 and others are the correlated time series which represent dew point (DEWP), temperature (TEMP), pressure (PRES), combined wind direction (CBWD), cumulated wind speed (LWS), cumulated hours of snow (LS) and cumulated hours of rain (LR), respectively. The notation definition is shown in Table 1.

Problem statement: Given a Multivariate time series set $X = \{X^{(1)}, X^{(2)}, \cdots, X^{(n)}\}$, we aim at forecasting the future measurements of a target time series in X. In general, the first time series $X^{(1)}$ is selected as the target forecasting time series. More specifically, we suppose that the target time series $X^{(1)}$ in X has measurements covering a window $\left[t_{a+1}, t_{a+l}\right]$ which includes l timestamps. Our goal is to predict the measurements of $X^{(1)}$ in the future window $\left[t_{a+l+1}, t_{a+l+p}\right]$ where p is the desirable horizon ahead of the current timestamp. This problem is called *p-steps ahead* forecasting.

IV. PROPOSED MODEL

The flowchart of the proposed model ANR-SAEs-VALSTM is presented in Figure 1. The original multivariate time series

TABLE 1. Notation definition.

Symbol	Description
X	$X = \{X^{(1)}, X^{(2)}, \cdots, X^{(n)}\}$ A set of multivariate time series
$X^{(i)}$	$X^{(i)} = \{x_1^{(i)}, x_2^{(i)}, \cdots, x_m^{(i)}\}, 1 \le i \le n$ A sequence of measurements from time series $X^{(i)}$
$x_t^{(i)}$	The measurement at time t of $X^{(i)}$, $1 \le t \le m$
Z	$Z = \{z_1, z_2, \cdots, z_p\}$ A sequence of predicted values
z_t	The measurement at time t of Z, $1 \le t \le p$
n	$n = \|X\|$ Number of multivariate time series
m	The length of time series
l	The length of model input
p	$p = \|Z\|$ Prediction size
K	Number of the decomposed IMFs
P	The demarcation point between high-frequency IMFs and low-frequency IMFs
N	The number of iterations of V-AdaBoost algorithm

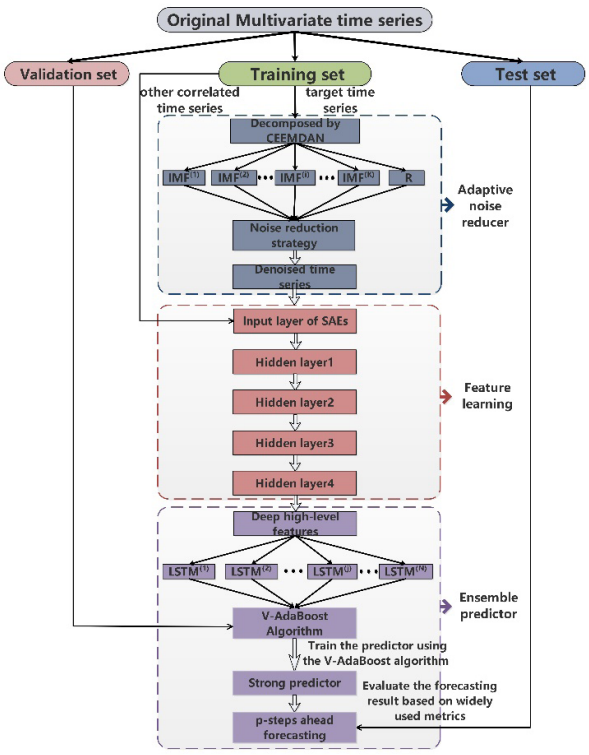


FIGURE 1. The framework of the proposed model (ANR-SAEs-VALSTM).

is divided into three parts. The first part is the training set, which is used to train the model and update parameters. The second part is the validation set, which is used to adjust the hyper-parameters of the model and adjust the weight of LSTM predictor obtained in each iteration of the V-AdaBoost algorithm. The last part is the test set which is applied to evaluate the prediction performance of the model. The model involves three stages. In the first stage, an Adaptive Noise Reducer (ANR) is applied to eliminate noise for the target time series of multivariate time series. In the second stage, we introduce an unsupervised feature extractor Stacked Auto-Encoders (SAEs) to capture the features of the de-noised multivariate time series. Finally, the de-noised features are fed into Validating AdaBoost (V-AdaBoost) algorithm based Long Short-term Memory (VALSTM) to train an over-fitting prevention ensemble predictor.

A. ADAPTIVE NOISE REDUCER

1) TIME SERIES DECOMPOSITION

In the first phase of the ANR, we apply the Complete Ensemble Empirical Mode Decomposition with Adaptive Noise (CEEMDAN) [42] to decompose time series into Multiple Intrinsic Mode Functions (IMFs) and a Residue (R). R has the lowest frequency and the frequency distribution of IMFs is from high to low. We assume that the original time series is $X = \{x_1, x_2, \cdots, x_m\}$, $E_j(\cdot)$ is the j-th IMF decomposed by EMD, w^i is the i-th Gaussian noise with $N(0, 1)$, and k is the serial number of IMF_k. The principles of decomposition is presented as follows:

Step 1: Add Gaussian noise $\varepsilon_0 w^i$ $(i = 1, \ldots I)$ to the original time series, and decompose $X + \varepsilon_0 w^i$ by EMD to get the first IMF:

$$IMF_1 = \frac{1}{I}\sum_{i=1}^{I} E_1(X + \varepsilon_0 w^i) \tag{1}$$

where ε_0 is an adaptive coefficient and I denotes the number of times the Gaussian noise added. In this paper, I is set to 100.

Step 2: When $k = 1$, calculate the first residue as $r_1 = X - IMF_1$. Then decompose $r_1 + \varepsilon_1 E_1(w^i)(i = 1, \ldots, I)$ by EMD to get the first EMD mode and define the second IMF as:

$$IMF_2 = \frac{1}{I}\sum_{i=1}^{I} E_1(r_1 + \varepsilon_1 E_1(w^i)) \tag{2}$$

Step 3: When $k = 2, \ldots, K$, calculate the k-th residue as $r_k = r_{k-1} - IMF_k$. Then decompose $r_k + \varepsilon_k E_k(w^i)$ $(i = 1, \ldots, I)$ by EMD to get the first EMD mode and define the $(k + 1)$-th IMF as:

$$IMF_{k+1} = \frac{1}{I}\sum_{i=1}^{I} E_1(r_k + \varepsilon_k E_k(w^i)) \tag{3}$$

Step 4: Repeat step3 to step4 until the residue cannot to be decomposed. The final residue satisfies:

$$R = X - \sum_{k=1}^{K} IMF_k \tag{4}$$

where K is the total number of IMFs, and the original time series can be expressed as:

$$X = \sum_{k=1}^{K} IMF_{\mathrm{k}} + \mathrm{R} \tag{5}$$

2) IMFS CLASSIFICATION

Permutation Entropy (PE) [43] is a kind of measure of time series complexity, and it is commonly used to discover complex structures from noisy time series. The size of the permutation entropy measures the randomness of the time series. The larger the permutation entropy is, the more noise the time series contains. In the second phase of ANR, the IMFs are divided into two categories: noisy and noise-free. The principle for IMFs classification is presented as follows:

Considering the time series $X = \{x_1, x_2, \cdots, x_m\}$, it is reconstructed into a matrix:

$$\begin{bmatrix} X(1) = \{x(1), x(1+\tau), \ldots, x(1+(d-1)\tau)\} \\ \vdots \\ X(j) = \{x(j), x(j+\tau), \ldots, x(j+(d-1)\tau)\} \\ \vdots \\ X(M) = \{x(M), x(M+\tau), \ldots, x(M+(d-1))\tau\} \end{bmatrix} \tag{6}$$

where d is the embedded dimension and τ is the embedded delay time. $M = m - (d-1)\tau$ is obtained from $m = M + (d-1)\tau$. Higher embedding dimensions can detect more complex patterns in time series, but the computation time will increase obviously. The embedded delay describes the time scale of complex structures. Empirically, in this paper, d is set to 4 and τ is set to 2.

Each row in the matrix is a reconstructed component, with a total of M. Then the elements of each component $X(j)$ are ranked in ascending order. It can be denoted as:

$$x[j+(j_1-1)\tau] \leq x[j+(j_2-1)\tau] \leq \cdots \leq x[j+(j_d-1)\tau] \tag{7}$$

where $j_1, j_2, \ldots, j_d$ represent the index number of the columns in the reconstructed component. Therefore, a set of symbols sequence can be obtained for each row in the matrix. The symbol sequence is recorded as:

$$S(l) = (j_1, j_2, \ldots, j_d) \tag{8}$$

where $l = 1, 2, \cdots, M$ and $M \leq d!$.

Calculate the probability distribution of the symbol sequences as $P_1, P_2, \cdots, P_M$, and then the PE value can be estimated according to the Shannon entropy:

$$H_p(d) = \frac{-\sum_{j=1}^{M} P_j ln p_j}{\ln(d!)} \tag{9}$$

where $ln(d!)$ is the normalization factor.

The IMFs classification is based on the PE value of each IMF. The threshold for PE value is set to 0.7 by literature review and multiple experiments. If the PE value of the IMF is greater than the threshold, the IMF is considered to contain noise, otherwise, the IMF is noise-free. Assuming P is the demarcation point between the noisy IMFs in high frequency and the noise-free IMFs in low frequency, the decomposed time series can be classified as:

$$X = \sum_{k=P}^{K} IMF^{(k)} + \sum_{k=1}^{P-1} IMF^{(k)} + R \tag{10}$$

3) HIGH-FREQUENCY IMFS NOISE REDUCTION

To de-noise the noisy IMFs in high frequency, the adaptive soft threshold λ and threshold function w_λ are defined as Eq. (11 - 12):

$$\lambda = \sigma * \sqrt{\frac{2 * lnm}{ln(K+1)}} \tag{11}$$

$$w_\lambda = \begin{cases} [sgn(w)](|w| - \lambda), & |w| \geq \lambda \\ 0, & |w| < \lambda \end{cases} \tag{12}$$

where σ, m and K is the standard deviation, length and the number of IMFs, respectively. w is a value in IMF. If the absolute value of w is greater than λ, takes $[sgn(w)](|w| -)\lambda$, otherwise, takes 0.

4) RECONSTRUCTION

The de-noised time series is obtained by constructed the de-noised IMFs in high frequency and the noise-free IMFs in low frequency. The de-noised time series is obtained as:

$$X = \sum_{k=1}^{P-1} IMF^{(k)} + \sum_{k=P}^{K} IMF^{'(k)} + R \tag{13}$$

where $IMF^{'(k)}(k = P, P+1, \cdots, K)$ is the de-noised IMF in high frequency and $IMF^{(k)}(k = 1, 2, \cdots, P-1)$ is the noise-free IMF in low frequency, and R is the residue.

B. UNSUPERVISED FEATURE LEARNING

In order to analyze the complex multivariate time series efficiently, an important method is to extract its features and the interdependence among variables. The method of feature extraction can be divided into supervised learning methods and unsupervised learning methods. The supervised learning method is a machine learning task that infers functions from a labeled training data set. However, time series is usually unlabeled. Converting it to labeled data is expensive, time-consuming, and requires professional knowledge. Another alternative is to use unsupervised learning method, it is capable of learning a layer of feature representations from unlabeled data. Furthermore, these layers of feature representations can be stacked to create deep networks. In our model, we use stacked auto-encoders (SAEs) [18]–[20] which is a kind of an unsupervised learning method for extracting the features of multivariate time series.

Auto-Encoder (AE) is a neural network that reconstructs the original input by encoding and decoding, as shown in Eq. (14) (15). In the encoding phase, AE obtains $a(X)$ by mapping the input vector X to the hidden layer. In the decoding phase, AE maps $a(X)$ to the reconstruction layer for reconstructing X. It can be considered that the hidden

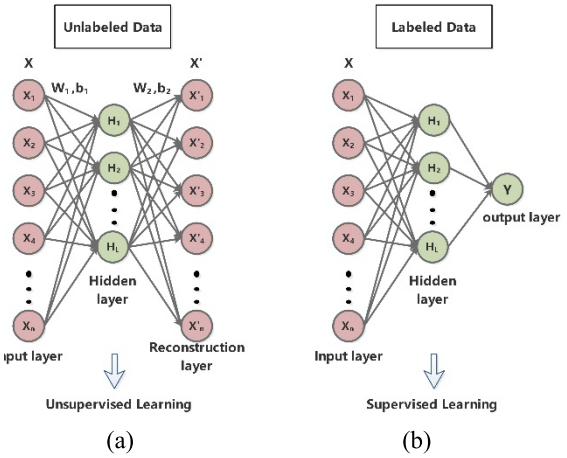


FIGURE 2. (a): Auto-encoder and (b): classical neural network.

layer $a(X)$ is an abstract feature representation of the input X when the input vector X and reconstruction vector $X^{'}$ are the same or similar.

$$a(X) = f(W_1 X + b_1) \tag{14}$$

$$X' = f(W_2 a(X) + b_2) \tag{15}$$

where W_1, W_2 denote the weight vector of the hidden layer and the reconstruction layer, respectively. b_1, b_2 denote the bias of the hidden layer and the reconstruction layer, respectively. f is an activation function.

The purpose of training a single layer AE is to find the optimal parameters W_1, W_2, b_1, and b_2 by minimizing the error between the input vector and the reconstruction vector. The error between the input vector and the output vector is expressed as the mean square error, as illustrated in Eq. (16).

$$L\left(x, x^{'}\right) = \frac{1}{mn}\sum_{i=1}^{n}\sum_{j=1}^{m}\left(x_{ij} - x_{ij}^{'}\right)^2 \tag{16}$$

where m represents the length of time series, and n represents the number of variables. The optimization function for minimizing the error between the input vector and the reconstruction vector is formulated as:

$$argmin_{W_1,W_2,b_1,b_2}\zeta = argmin_{W_1,W_2,b_1,b_2}L\left(x, x'\right) \tag{17}$$

Figure 2 (a) shows the structure of an auto-encoder with n input units and L hidden units. n corresponds to variables number of multivariate time series and L corresponds to the dimensions of the extracted features. Training AE is an unsupervised learning task because the labels are not given. The advantage of unsupervised learning is that the feature learning and time series prediction can be separated, which transforms the model into a multi-task learning model. Figure 2 (b) demonstrates a classical neural network, which is a supervised learning task and labels are given to minimize the error of predicting labels.

Single-layer AE can only construct simple function mappings, making it difficult to handle complex multivariate nonlinear time series. Therefore, Stacked Auto-Encoders (SAEs) is constructed by stacking several single-layer AEs to solve this problem. In SAEs, the single-layer AE maps the input into the first hidden layer. The reconstruction layer of the first single-layer AE is abandoned after training the first single-layer AE, the hidden layer becomes the input layer of the second single-layer AE, and the same for other layers. Each single-layer AE is trained by using the optimization function as formulated in the Eq. (17) and the same gradient descent method. Figure 3 presents the SAEs used in our model. The number of input units is n, which is determined by the number of variables in the multivariate time series. The depth of SAEs is set to 5 and the number of units in each hidden layer is set to 12, 10, 8 and 6, respectively by trial and error.

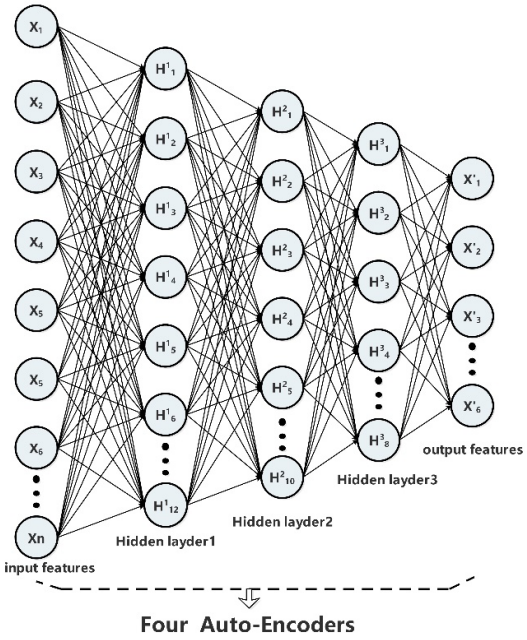


FIGURE 3. The structure of stack auto-encoders (SAEs).

C. OVER-FITTING PREVENTION ENSEMBLE PREDICTOR

1) LONG SHORT-TERM MEMORY

Long Short-Term Memory (LSTM) is a kind of variant of Recurrent Neural Network (RNN). The threshold structure of LSTM can solve the gradient vanishing and gradient exploding problems. LSTM uses memory blocks to replace the traditional neurons in the hidden layer. As Figure 4 presents, the LSTM block contains a memory cell (C_t), an input gate (i_t), an output gate (o_t), and a forget gate (f_t). $W_i, W_f, W_c, W_o, U_i, U_f, U_c$and U_o are weight matrixes. $\otimes$ is the point-wise multiplication. *sigmoid* activation function and *tanh* activation function for gating are marked as σ and *tanh*, respectively. At time t, x_t denotes the input and h_t the hidden state. $\hat{C}_t$ is the candidate state of the memory cell, which determines how much the input is received in the cell state. The calculations for each gate, input candidate,

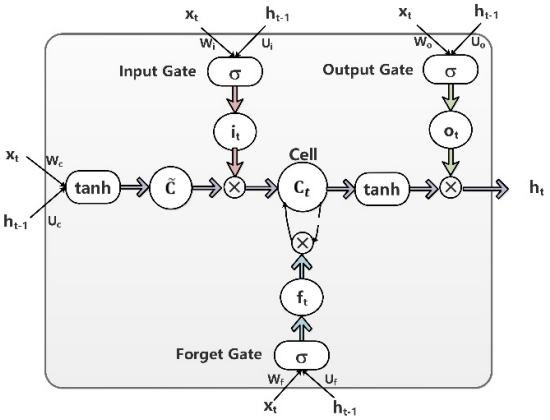


FIGURE 4. The structure of an LSTM memory block.

hidden state, and cell state are presented as following:

$$i_t = \sigma\left(W_i * x_t + U_i * h_{t-1} + b_i\right) \tag{18}$$

$$\tilde{C}_t = \tan\left(W_c * x_t + U_c * h_{t-1} + b_c\right) \tag{19}$$

$$f_t = \sigma\left(W_f * x_t + U_f * h_{t-1} + b_f\right) \tag{20}$$

$$C_t = i_t * \tilde{C}_t + f_t * C_{t-1} \tag{21}$$

$$o_t = \sigma\left(W_o * x_t + U_o * h_{t-1} + b_o\right) \tag{22}$$

$$h_t = o_t * tan(C_t) \tag{23}$$

2) THE PROPOSED ENSEMBLE PREDICTOR

AdaBoost is an algorithm that aims to combine a series of weak learners to a strong learner. Although AdaBoost algorithm has achieved satisfactory results on the classification problem, few works of literature apply it to time series forecasting. There are two problems using the AdaBoost algorithm for time series forecasting. One is how to determine the update of the weight distribution of samples, and the other is how to assign weight to each weak predictor. To the best of our knowledge, current AdaBoost algorithms are all based on the training set, and both the weight distribution of the samples and the weight of each weak predictor are determined by the prediction error on the training set. As the number of iterations of the AdaBoost algorithm increases, the upper bound of the prediction error on the training set will continually drop, and the risk of over-fitting will inevitably occur.

To solve the over-fitting problem mentioned above, in this study, Validating AdaBoost algorithm (V-AdaBoost) is proposed to combine weak LSTM predictors for developing a strong predictor which is called VALSTM. The V-AdaBoost algorithm adds the weight distribution of the validation set to the iterative process. The weight distribution of the samples and the weight of each weak predictor are determined not only by the prediction error on the training set but also the prediction error on the validation set. The prediction error on the validation set plays an out-of-sample evaluation on the LSTM predictor. If the error on the validation set is small, it has a positive effect on the weight of the LSTM predictor, otherwise, it has a negative effect, which prevents over-fitting to a great extent and enhances the generalization capability of the LSTM predictor. The principle of the V-AdaBoost algorithm is described below.

The original time series $X = \{x_1, x_2, \cdots, x_m\}$ is divided into three parts, i.e., training set $Train = \{x_1, x_2, \cdots, x_{l_1}\}$, validation set $Validate = \left\{x_{l_1+1}, x_{l_1+2}, \cdots, x_{l_1+l_2}\right\}$, and test set $Test = \{x_{l_1+l_2+1}, x_{l_1+l_2+2}, \cdots, x_m\}$. The training set and the validation set are fed into VALSTM to train an ensemble predictor. The test set is used to evaluate the prediction performance. Initialize the weight distribution of the training set and the validation set to $DT_0 = \{\frac{1}{l_1}, \frac{1}{l_1}, \cdots, \frac{1}{l_1}\}$ and $DV_0 = \{\frac{1}{l_2}, \frac{1}{l_2}, \cdots, \frac{1}{l_2}\}$, respectively. The weak LSTM predictor in the k-th iteration is defined as $\xi_k(\cdot)$, and a total of N iterations. The discriminative prediction error E_k in the k-th iteration of the V-AdaBoost algorithm can be denoted as:

$$E_k = \frac{1}{2}\left(\frac{l_1}{l_1+l_2}\sum_{i=1}^{l_1} DT_k^i J\left(e_k^i - \theta\right) + \frac{l_2}{l_1+l_2}\sum_{i=l_1+1}^{l_1+l_2} DV_k^i J\left(e_k^i - \theta\right)\right) \tag{24}$$

where DT_k^i and DV_k^i represent the weight of the i-th sample of the training set and validating set, respectively. E_k is denoted as the sum of the discriminative prediction error on the training set and the validation set. As the length of the training set and validation set are different, $\frac{l_1}{l_1+l_2}$ and $\frac{l_2}{l_1+l_2}$ are used to represent the proportion of them. e_k^i denotes the prediction error on the i-th sample in the k-th iteration of the V-AdaBoost algorithm, and it can be calculated by $e_k^i = \frac{|\xi_k(x_i) - x_i|}{x_i}$. θ is the threshold of the prediction error and $J(\cdot)$ is defined as a discriminative function:

$$J(x) = \begin{cases} 1, & if\ x > 0 \\ -1, & otherwise \end{cases} \tag{25}$$

which means the case that e_k^i exceeds θ will take positive effect, vice versa.

Then, in order to ensure that the selected LSTM predictor in each iteration has good prediction performance and is not over-fitting, a double conditional discriminative function is presented as:

$$\psi = \begin{cases} True, & if\ E_k \le \frac{1}{2}\ and\ \frac{\eta_V}{\eta_T} < 1 \\ False, & otherwise \end{cases} \tag{26}$$

where η_T and η_V are the average prediction errors on the training set and the validation set respectively, which can be obtained by $\eta_T = \frac{\sum_{i=1}^{l_1} J(e_k^i - \theta)}{l_1}$ and $\eta_V = \frac{\sum_{i=l_1+1}^{l_1+l_2} J(e_k^i - \theta)}{l_2}$ respectively. $E_k \le \frac{1}{2}$ indicates that the discriminative prediction error is less than 50%, which ensures the good prediction performance of the selected LSTM predictor. $\frac{\eta_V}{\eta_T} < 1$ means that the average prediction error on the validation set is smaller than the average prediction error on the training set,

which ensures that the selected LSTM predictor is not over-fitting.

The importance of the selected LSTM predictor in the k-th iteration can be denoted as:

$$\alpha_k = \frac{1}{2}\ln\frac{1-E_k}{E_k} \tag{27}$$

which means the case that the smaller discriminative prediction error of the selected LSTM predictor achieves, the more important in the final ensemble strong predictor.

After the end of the k-th iteration, the new weight distributions of the training set and the validation set are updated to:

$$DT^i_{k+1} = \frac{DT^i_k}{ZT_k} e^{\alpha_k J(e^i_k-\theta)} \tag{28}$$

$$DV^i_{k+1} = \frac{DV^i_k}{ZV_k} e^{\alpha_k J(e^i_k-\theta)} \tag{29}$$

where ZT_k and ZV_k are normalization factors, which can be denoted as $ZT_k = \sum_{i=1}^{l_1} DT^i_k e^{\alpha_k J(e^i_k-\theta)}$ and $ZV_k = \sum_{i=l_1+1}^{l_1+l_2} DV^i_k e^{\alpha_k J(e^i_k-\theta)}$. The larger the discriminative prediction error of the sample achieves, the larger the new weight it obtains after updating, vice versa.

After N iterations by V-AdaBoost algorithm, N weak LSTM predictors are trained and different weights are assigned to them. Finally, the ensemble strong predictor can be obtained by:

$$P_{final} = \sum_{k=1}^{N} w_k \xi_k \tag{30}$$

where $w_k = \frac{\alpha_k}{\sum_{k=1}^{N}\alpha_k}$ is the normalized weight.

Since the weight distribution of the validation set is added to the V-AdaBoost algorithm, both the computational complexity and space complexity of the V-AdaBoost algorithm are increased. The computational complexity of the V-AdaBoost algorithm is $O(N(T + 3l_1 + 3l_2) + 2N)$, where T is the average time for the V-AdaBoost algorithm to train the LSTM predictor in each iteration, $3l_1$ is the time consumption for calculating the loss on the training set and updating the weight distribution of the training set, $3l_2$ is the time consumption for calculating the loss on the validation set and updating the weight distribution of the validation set, and $2N$ is the time consumption for calculating the final strong predictor. The computational complexity of the original AdaBoost algorithm is $O(N(T + 3l_1) + 2N)$ because it does not consider the weight distribution of the validation set. It can be seen that the computational complexity is just slightly increased. Correspondingly, the space complexity is also slightly increased because the size of the validation set is much smaller than the size of the training set.

The V-AdaBoost algorithm can be summarized as follows:

Algorithm 1 The V-AdaBoost algorithm

Input:

The original time series $X = \{x_1, x_2, \cdots, x_m\}$;
The threshold of prediction error θ;
The number of iterations N;

Output:

The ensemble strong predictor P_{final}

1: Divide the time series into three parts, *i.e.*, the training set $\{x_1, x_2, \cdots, x_{l_1}\}$, the validation set $\{x_{l_1+1}, x_{l_1+2}, \cdots, x_{l_1+l_2}\}$ and the test set $\{x_{l_1+l_{2+1}}, x_{l_1+l_{2+2}}, \cdots, x_m\}$

2: Initialize the weight distribution of the training set $DT_0 = \{\frac{1}{l_1}, \frac{1}{l_1}, \cdots \frac{1}{l_1}\}$ and validation set $DV_0 = \{\frac{1}{l_2}, \frac{1}{l_2}, \cdots \frac{1}{l_2}\}$

3: **while** k ≤ N **do**:

4: Train a LSTM predictor with the current weight distribution of the training set and validation set as the k-th base LSTM predictor ξ_k

5: Calculate the loss $e^i_k = \frac{|\xi_k(x_i)-x_i|}{x_i}$

6: Calculate the total loss

$$E_k = \frac{1}{2}\left(\frac{l_1}{l_1+l_2}\sum_{i=1}^{l_1} DT^i_k J\left(e^i_k-\theta\right) + \frac{l_2}{l_1+l_2}\sum_{i=l_1+1}^{l_1+l_2} DV^i_k J\left(e^i_k-\theta\right)\right)$$

7: Calculate the average loss on the training set and the validation set

$$\eta_T = \frac{\sum_{i=1}^{l_1} J(e^i_k-\theta)}{l_1}, \quad \eta_V = \frac{\sum_{i=l_1+1}^{l_1+l_2} J(e^i_k-\theta)}{l_2}$$

8: **if not** $E_k \le \frac{1}{2}$ *and* $\frac{\eta_V}{\eta_T} < 1$:
return to step4

9: Calculate the weight of the$k$-th base predictor $\xi_{\mathbf{k}}$
$\alpha_k = \frac{1}{2}\ln\frac{1-E_k}{E_k}$

10: update the weight distribution
$DT^i_{k+1} = \frac{DT^i_k}{ZT_k} e^{\alpha_k J(e^i_k-\theta)}$, $DV^i_{k+1} = \frac{DV^i_k}{ZV_k} e^{\alpha_k J(e^i_k-\theta)}$
where $ZT_k = \sum_{i=1}^{l_1} DT^i_k e^{\alpha_k J(e^i_k-\theta)}$ and
$ZV_k = \sum_{i=l1+1}^{l_1+l_2} DV^i_k e^{\alpha_k J(e^i_k-\theta)}$

11: **end while**

12: Calculate the normalized weight distribution of the final predictor

$$w_k = \frac{\alpha_k}{\sum_{k=1}^{N}\alpha_k}, \text{ where } k = 1, 2, \cdots, N$$

13: Calculate the final hypothesis

$$P_{final} = \sum_{k=1}^{N} w_k \xi_k$$

V. EXPERIMENTS AND ANALYSIS

A. DATA DESCRIPTION

In this section, we conducted extensive experiments with 6 models (including the proposed model) on 2 benchmark datasets for multivariate time series forecasting. Furthermore, to demonstrate the efficiency of our model design, an ablation study is conducted. Finally, we vary two problem parameters in the experiment, including the number of variables $|X|$ and the input length l of the time series when training the model.

Two publicly available benchmark datasets are used in our experiment. The datasets details are summarized in Table 2.

Beijing PM2.5 dataset [44] contains hourly PM2.5 data and the correlated variables in Beijing of China. PM2.5 is the

TABLE 2. Dataset details, where T is the length of time series, D is the total number of variables, and L is the sample rate.

Datasets	T	D	L
Beijing PM2.5	43824	8	1 hour
GEFCom2014 Electricity Price	21552	3	1 hour

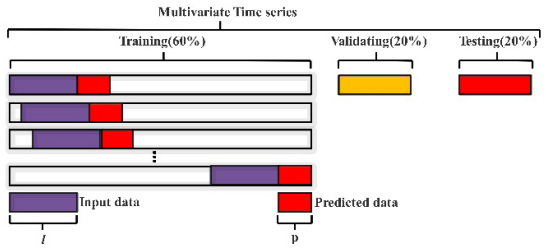


FIGURE 5. Training, validating and testing.

target time series. The correlated variables include dew point (DEWP), temperature (TEMP), pressure (PRES), combined wind direction (CBWD), cumulated wind speed (LWS), cumulated hours of snow (LS), and cumulated hours of rain (LR). Totally, the size of Beijing PM2.5 dataset is 43824.

GEFCom2014 Electricity Price datasets [45] were published in the Global Energy Forecasting Competition 2014. The datasets contain four parts, which are electric load dataset, electricity price dataset, wind power dataset, and solar power dataset, respectively. In our study, the electricity price dataset is selected for multivariate time series forecasting. The target time series is hourly locational marginal price, and the correlated variables include zonal load and system load. Totally, the size of GEFCom2014 Electricity Price dataset is 21552.

B. EXPERIMENTAL DESIGN

In our experiment, each dataset is split into a training set (60%), a validation set (20%), and a test set (20%) in chronological order, as illustrated in Figure 5. When training the proposed model, the training data are segmented into multiple training cases using a sliding window. In particular, in each segment, a sequence of l samples are used as the input data to the proposed model and the immediately following p samples are used as the actual target data. The validation data are used to adjust the hyper-parameters of the model and adjust the weight of LSTM predictor obtained in each iteration of V-AdaBoost algorithm. The test data are used to evaluate the model.

C. METRICS

Two conventional metrics which are defined as follows are used to evaluate the prediction performance.

- Root Mean Squared Error (RMSE)

$$\mathrm{RMSE} = \sqrt{\frac{1}{m}\sum_{t=1}^{m}(z_t - x_t)^2} \tag{31}$$

- Mean Absolute Error (MAE)

$$\mathrm{MAE} = \frac{1}{m}\sum_{t=1}^{m}|z_t - x_t| \tag{32}$$

where x_t and z_t represent the actual value and prediction value at time t in the test set, respectively. m is the length of the test set. RMSE and MAE measure the prediction error, and the lower value is better.

D. EXPERIMENT 1: MODELS FOR COMPARISON

To demonstrate the superiority of the proposed model ANR-SAEs-VALSTM, we compare it against the following 5 models:

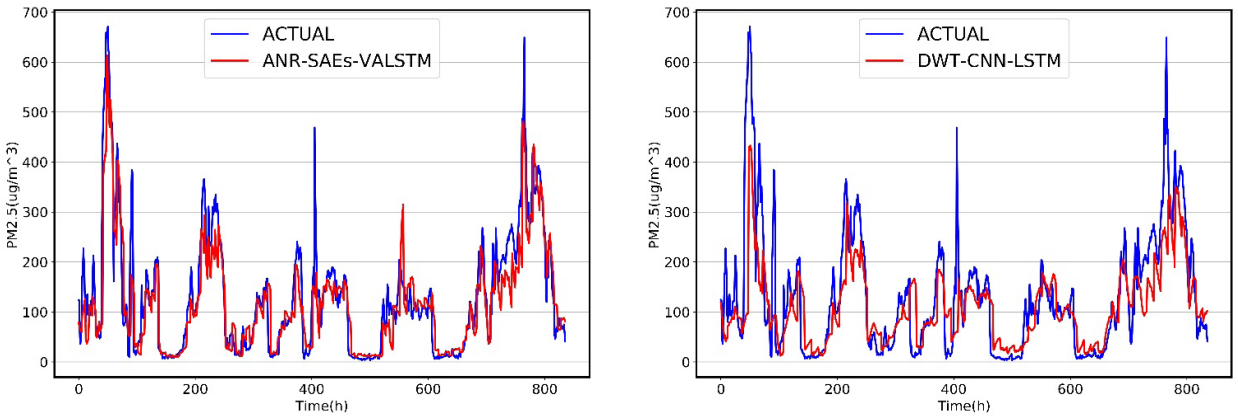
- ARIMA is the Autoregressive Moving Average Model [27].
- LSVR is the Vector Autoregression (VAR) model with Support Vector Regression objective function [33].
- GP is the Gaussian Process for time series modeling [36].
- WSAEs-LSTM uses the Discrete Wavelet Transform (DWT) to decompose time series to eliminate noise. Then SAEs is applied to extract high-level features. Finally, the high-level de-noised features are fed into LSTM for time series forecasting [9].
- DWT-CNN-LSTM uses DWT to decompose the original time series into several subsequences. Then CNN is used to learn the abstract feature representation of each subsequence. Finally, the features are fed into LSTM for time series forecasting [40].

The experiment uses grid search over tunable hyper-parameters for each model and dataset. Specifically, for ARIMA, the number of AR parameters p and MA parameters q are determined by the minimum AIC of each input sample, and the difference order d is determined by Dickey-Fuller test. For LSVR, the regularization coefficient λ is set from $\{2^{-10}, 2^{-8}, \cdots, 2^{8}, 2^{10}\}$. For GP, the noise level α and the RBF kernel bandwidth σ are set from $\{2^{-10}, 2^{-8}, \cdots, 2^{8}, 2^{10}\}$. For WSAEs-LSTM, the decomposition layer of DWT is set from {1, 2, 3, 4}, the size of the hidden layer of SAEs is set to 10, and the hidden layer of LSTM is set from {50, 100, 150}. For DWT-CNN-LSTM, the decomposition layer of DWT is set from {1, 2, 3, 4}, the convolution kernel of CNN is set to 3×3, the pooling layer is set to maxpooling1d, and the hidden dimension of LSTM is set from {50, 100, 150}. For the proposed model, the threshold of permutation entropy is set to 0.7, the depth of SAEs is set to 5, the hidden dimensions of SAEs are set to {12, 10, 8, 6}, respectively, the number of LSTM layer is set from {1, 2, 3}, and the hidden dimension of LSTM is set from {50, 100, 150}.

The evaluation results of all models in all metrics are summarized in Table 3. The horizon is set to from {3, 6, 12, 24}. The larger the horizon is, the harder the forecasting task is because of the larger uncertainty. The best result for each dataset and horizon is highlighted. ANR-SAEs-VALSTM

TABLE 3. Results summary of all models on two datasets.

Dataset		Beijing PM2.5				GEFCom2014 Electricity Price			
		horizon (p)				horizon (p)			
Model	Metrics	3	6	12	24	3	6	12	24
ARIMA	MAE	26.51	35.59	52.54	70.89	8.31	10.34	12.61	14.79
	RMSE	42.07	54.82	73.31	94.30	15.12	17.31	19.35	20.84
LSVR	MAE	24.28	35.93	48.65	60.32	7.52	8.67	9.40	9.80
	RMSE	39.28	55.13	70.86	83.38	13.20	14.91	15.74	16.43
GP	MAE	23.89	36.98	51.84	64.75	12.33	15.33	14.75	12.13
	RMSE	40.06	58.58	77.24	91.03	22.91	26.99	26.66	23.05
WSAEs-LSTM	MAE	26.21	35.68	49.77	59.09	5.53	7.74	9.19	10.39
	RMSE	36.91	52.63	69.10	84.84	9.71	14.04	15.85	17.54
DWT-CNN-LSTM	MAE	23.58	32.36	52.53	60.59	5.45	6.69	8.52	11.70
	RMSE	37.95	49.65	70.82	85.87	9.39	11.65	14.39	18.29
ANR-SAEs-VALSTM	MAE	**20.73**	**30.85**	**40.82**	**48.47**	**4.94**	**5.99**	**7.53**	**9.09**
	RMSE	**35.45**	**47.06**	**62.57**	**77.18**	**8.88**	**10.35**	**13.06**	**16.28**



(a) (b)

FIGURE 6. The prediction results (red) by ANR-SAEs-VALSTM (a) and by DWT-CNN-LSTM (b) vs. the actual data (blue) on Beijing PM2.5 dataset with horizon = 24.

outperforms the best comparative model DWT-CNN-LSTM on different datasets and in different horizons. In particular, ANR-SAEs-VALSTM is 12.09%, 4.67%, 22.29%, and 20% better than DWT-CNN-LSTM with respect to MAE metric when the horizons are 3, 6, 12, and 24, respectively. Besides, ANR-SAEs-VALSTM also performs well on the GEFCom2014 Electricity Price dataset, and is 9.36%, 10.46%, 11.62%, and 22.31% better than DWT-CNN-LSTM with respect to MAE metric when the horizons are 3, 6, 12, and 24, respectively.

For visualization, taking the (horizon = 24) as an example, we show the prediction result of ANR-SAEs-VALSTM and DWT-CNN-LSTM (the best comparative model) on the Beijing PM2.5 dataset and GEFCom2014 Electricity Price dataset in Figure 6 and 7. For clarity, only 10% of the prediction results are shown. The results show that ANR-SAEs-VALSTM generally fits the actual data much better than DWT-CNN-LSTM, which demonstrates that the ANR-SAEs-VALSTM has better prediction performance.

E. EXPERIMENT 2: ABLATION STUDY

To demonstrate the efficiency of the proposed model design, an ablation study is conducted. Specifically, we remove each component by one at a time in ANR-SAEs-VALSTM. The models without different components are defined as follows:

- ANR-SAEs-ALSTM:ANR-SAEs-VALSTM with the normal AdaBoost algorithm.
- ANR-SAEs-δ ALSTM: ANR-SAEs-VALSTM with the δ-agree AdaBoost algorithm [20].
- SAEs-VALSTM:ANR-SAEs-VALSTM without the de-noising component (ANR).
- ANR-VALSTM:ANR-SAEs-VALSTM without the features extractor component (SAEs).

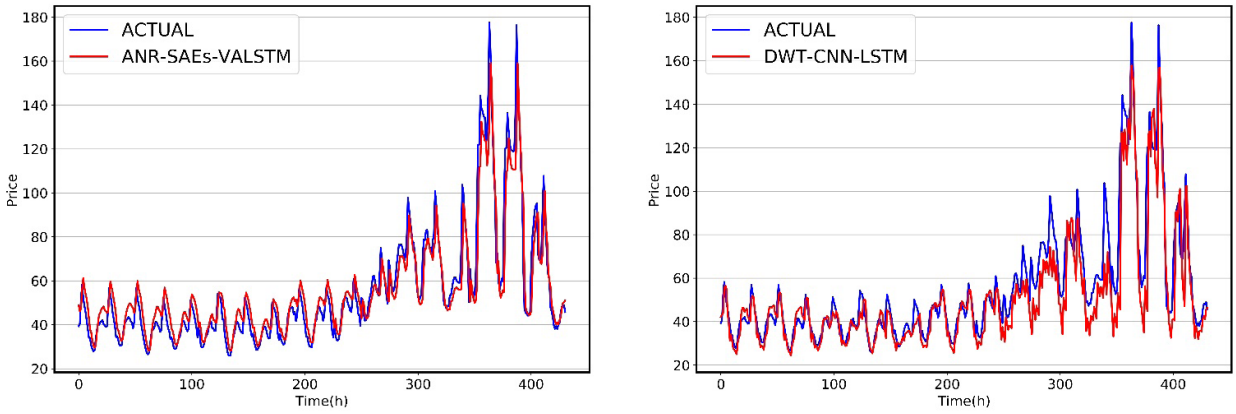


FIGURE 7. The prediction results (red) by ANR-SAEs-VALSTM (a) and by DWT-CNN-LSTM (b) vs. the actual data (blue) on GEFCom2014 Electricity Price dataset with horizon = 24.

TABLE 4. Results summary in ablation study.

Dataset		Beijing PM2.5			
		horizon (p)			
Model	Metrics	3	6	12	24
SAEs-VALSTM	MAE	25.05	34.18	48.17	55.23
	RMSE	36.61	49.80	67.44	82.55
ANR-VALSTM	MAE	27.19	36.17	48.20	55.63
	RMSE	38.25	50.64	66.68	82.76
ANR-SAEs-ALSTM	MAE	23.16	33.58	44.90	51.33
	RMSE	35.53	49.27	65.18	81.80
ANR-SAEs-δALSTM	MAE	23.05	32.43	42.11	49.62
	RMSE	35.49	48.67	63.52	79.19
ANR-SAEs-VALSTM	MAE	**20.73**	**30.85**	**40.82**	**48.47**
	RMSE	**35.45**	**47.06**	**62.57**	**77.18**

The experiment takes the Beijing PM2.5 dataset as an example, and the horizons are from 3 to 24. Table 4 and Figure 8 shows the prediction results of different models. Several observations for these results are worth highlighting:

- The best results are obtained by ANR-SAEs-VALSTM in all horizons.
- Removing the ANR component or SAEs component caused the most significant performance drops in all horizons, showing the crucial role of ANR component and SAEs component.
- ANR-SAEs-VALSTM outperforms ANR-SAEs-ALSTM with respect to MAE and RMSE, which demonstrates that the proposed V-AdaBoost algorithm is superior to the normal AdaBoost algorithm.
- ANR-SAEs-VALSTM has better prediction performance than ANR-SAEs-δALSTM with respect to MAE and RMSE, which demonstrates that the proposed V-AdaBoost algorithm is superior to the δ-agree AdaBoost algorithm.

The conclusion is that the architecture design of the proposed model is the most robust across all experiment settings mentioned above. The ANR component plays a crucial role in the proposed model, this is because ANR eliminates the noise of the original time series which improves the efficiency and effectiveness of the subsequent analysis. Besides, SAEs learns deep features and provides LSTM with a more reasonable data input format, which reduces the training time and speeds up the convergence of LSTM. What's more, as illustrated in Figure 8 (b), the ANR-SAEs-ALSTM with the normal AdaBoost algorithm performs well when the horizon is 3 and 6, but not good in large horizons. In contrast, ANR-SAEs-VALSTM with the V-AdaBoost algorithm has a greater advantage when the horizon is large.

F. EXPERIMENT 3: PARAMETERS SELECTION

The number of variables $|X|$ and the length l of input time series when training the model have a great impact on the prediction performance of ANR-SAEs-VALSTM. In this section, the experiment takes the Beijing PM2.5 dataset as

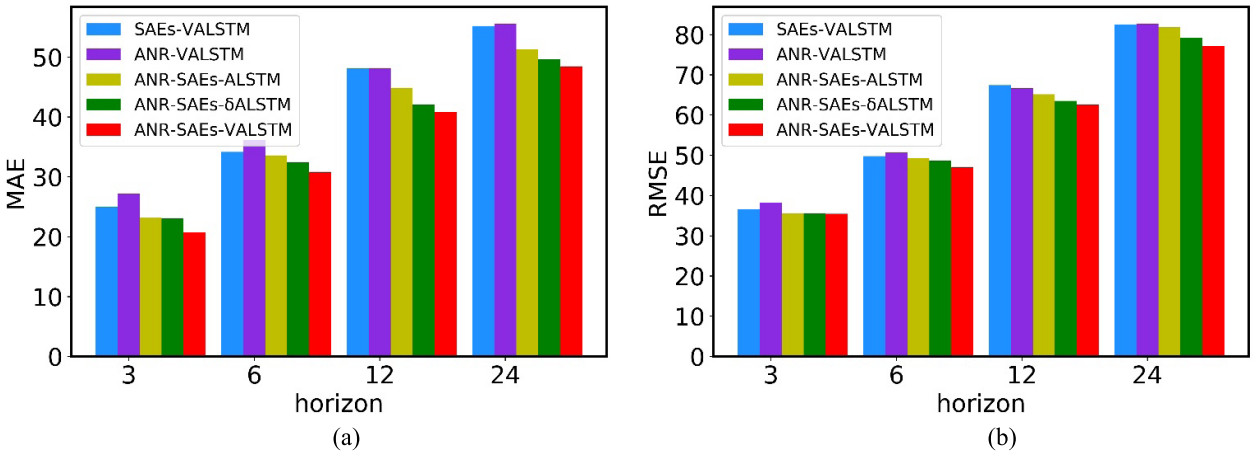


FIGURE 8. Results summary in Ablation Study, MAE (a) and RMSE (b).

TABLE 5. Result of MAE and RMSE when varying $|X|$ and l.

Dataset			Beijing PM2.5			
horizon (p)			6			
the input length			l			
X	$\|X\|$	Metrics	3	6	12	24
PM2.5	1	MAE	40.76	36.91	38.40	42.16
		RMSE	53.58	55.37	53.97	63.55
PM2.5, DEWP	2	MAE	36.67	35.31	36.75	36.91
		RMSE	49.82	51.22	55.06	53.47
PM2.5, DEWP, TEMP, PRES	4	MAE	34.48	33.06	34.90	35.62
		RMSE	50.86	49.46	50.60	52.49
PM2.5, DEWP, TEMP, PRES, CBWD, LWS	6	MAE	33.06	32.41	33.33	34.18
		RMSE	48.15	49.93	50.56	49.74
PM2.5, DEWP, TEMP, PRES, CBWD, LWS, LS, LR	8	MAE	34.97	31.86	**30.85**	32.16
		RMSE	52.06	48.96	**47.06**	50.58

an example and sets the horizon to 6. Table 5 shows the prediction results of ANR-SAEs-VALSTM while the number of variables $|X|$ is set from $\{1, 2, 4, 6, 8\}$ and the input length l is set from $\{3, 6, 12, 24\}$.

When l is very short, e.g., $l = 3$, ANR-SAEs-VALSTM has poor prediction performance, which suggests that it is hard to learn enough historical features and make an accurate prediction. Besides, the prediction performance of ANR-SAEs-VALSTM significantly drops when l is very long, e.g., $l = 24$. This is because the learned features are too redundant to predict correctly. In contrast, ANR-SAEs-VALSTM has the smallest prediction error when $l = 12$. As $|X|$ increases, the prediction performance of ANR-SAEs-VASLTM also increases generally, which means that there are strong dependencies among variables and extracting the features of them is capable of enhancing the prediction ability.

VI. CONCLUSION

Multivariate time series forecasting has a great significance across many domains. In this paper, aiming at improving the prediction performance of multivariate time series with noise and subsequences with strong nonlinear fluctuations. We have proposed a novel ensemble model called ANR-SAEs-VALSTM. The Adaptive Noise Reducer (ANR) is introduced for time series noise elimination. To further enhance the forecasting performance, SAEs is applied to capture the dependencies among multivariate time series and an over-fitting prevention ensemble algorithm (V-AdaBoost) is introduced to combine LSTM predictors to obtain a strong predictor. Extensive experiments on the Beijing PM2.5 dataset and GEFCom2014 electricity Price dataset demonstrate that the proposed model outperforms other models for multivariate time series forecasting.

However, there are missing data at some timestamps in the time series dataset. At present, we just replace them with nearby data, which may reduce the prediction performance of the model. Our further work will focus on finding a more effective method to solve this problem.

REFERENCES

[1] B. Wang, H. Huang, and X. Wang, "A novel text mining approach to financial time series forecasting," *Neurocomputing*, vol. 83, pp. 136–145, Apr. 2012.

[2] A. M. Rather, A. Agarwal, and V. N. Sastry, "Recurrent neural network and a hybrid model for prediction of stock returns," *Expert Syst. Appl.*, vol. 42, no. 6, pp. 3234–3241, 2015.

[3] A. Gautam and V. Singh, "A novel approach for decomposition of financial time series," in *Proc. Int. Conf. IEEE Recent Innov. Signal Process. Embedded Syst. (RISE)*, Oct. 2017, pp. 537–542.

[4] H. Liu, X.-W. Mi, and Y.-F. Li, "Wind speed forecasting method based on deep learning strategy using empirical wavelet transform, long short term memory neural network and elman neural network," *Energy Convers. Manage.*, vol. 156, pp. 498–514, Jan. 2018.

[5] F. Wang, Y. Yu, Z. Zhang, J. Li, Z. Zhen, and K. Li, "Wavelet decomposition and convolutional LSTM networks based improved deep learning model for solar irradiance forecasting," *Appl. Sci.*, vol. 8, no. 8, p. 1286, 2018.

[6] R.-G. Cirstea, D.-V. Micu, G.-M. Muresan, C. Guo, and B. Yang. (2018). "Correlated time series forecasting using deep neural networks: A summary of results." [Online]. Available: https://arxiv.org/abs/1808.09794

[7] H. Hassani, A. Dionisio, and M. Ghodsi, "The effect of noise reduction in measuring the linear and nonlinear dependency of financial markets," *Nonlinear Anal., Real World Appl.*, vol. 11, no. 1, pp. 492–502, 2010.

[8] J. Gao, H. Sultan, J. Hu, and W.-W. Tung, "Denoising nonlinear time series by adaptive filtering and wavelet shrinkage: A comparison," *IEEE Signal Process. Lett.*, vol. 17, no. 3, pp. 237–240, Mar. 2010.

[9] W. Bao, J. Yue, and Y. Rao, "A deep learning framework for financial time series using stacked autoencoders and long-short term memory," *PLoS ONE*, vol. 12, no. 7, 2017, Art. no. e0180944.

[10] M. Xu, M. Han, and H. Lin, "Wavelet-denoising multiple echo state networks for multivariate time series prediction," *Inf. Sci.*, vol. 465, pp. 439–458, Oct. 2018.

[11] S. Farahmand, T. Sobayo, and D. J. Mogul, "Noise-assisted multivariate EMD-based mean-phase coherence analysis to evaluate phase-synchrony dynamics in epilepsy patients," *IEEE Trans. Neural Syst. Rehabil. Eng.*, vol. 26, no. 12, pp. 2270–2279, Dec. 2018.

[12] F. Y. Abdalla, Y. Zhao, and L. Wu, "Denoising ECG signal by complete EEMD adaptive noise," in *Proc. IEEE Int. Symp. Signal Process. Inf. Technol. (ISSPIT)*, Dec. 2017, pp. 337–342.

[13] L. Bai, Z. Han, Y. Li, and S. Ning, "A hybrid de-noising algorithm for the gear transmission system based on CEEMDAN-PE-TFPF," *Entropy*, vol. 20, no. 5, p. 361, 2018.

[14] B. Fadlallah, B. Chen, A. Keil, and J. Príncipe, "Weighted-permutation entropy: A complexity measure for time series incorporating amplitude information," *Phys. Rev. E, Stat. Phys. Plasmas Fluids Relat. Interdiscip. Top.*, vol. 87, no. 2, 2013, Art. no. 022911.

[15] S.-Y. Lin, C.-C. Chiang, J.-B. Li, Z.-S. Hung, and K.-M. Chao, "Dynamic fine-tuning stacked auto-encoder neural network for weather forecast," *Future Gener. Comput. Syst.*, vol. 89, pp. 446–454, Dec. 2018.

[16] U. Pak, C. Kim, U. Ryu, K. Sok, and S. Pak, "A hybrid model based on convolutional neural networks and long short-term memory for ozone concentration prediction," *Air Qual., Atmos. Health*, vol. 11, no. 8, pp. 883–895, 2018.

[17] N. Pang, F. Yin, X. Zhang, and X. Zhao, "A robust approach for multivariate time series forecasting," in *Proc. 8th Int. Symp. Inf. Commun. Technol.*, 2017, pp. 106–113.

[18] K. Chen, J. Hu, and J. He, "A framework for automatically extracting overvoltage features based on sparse autoencoder," *IEEE Trans. Smart Grid*, vol. 9, no. 2, pp. 594–604, Mar. 2018.

[19] F. Lv, C. Wen, M. Liu, and Z. Bao, "Weighted time series fault diagnosis based on a stacked sparse autoencoder," *J. Chemometrics*, vol. 31, no. 9, p. e2912, 2017.

[20] T. Zhou *et al.*, "σ-agree AdaBoost stacked autoencoder for short-term traffic flow forecasting," *Neurocomputing*, vol. 247, pp. 31–38, Jul. 2017.

[21] H. Y. Kim and C. H. Won, "Forecasting the volatility of stock price index: A hybrid model integrating LSTM with multiple GARCH-type models," *Expert Syst. Appl.*, vol. 103, pp. 25–37, Aug. 2018.

[22] Y.-L. Hu and L. Chen, "A nonlinear hybrid wind speed forecasting model using LSTM network, hysteretic ELM and differential evolution algorithm," *Energy Convers. Manage.*, vol. 173, pp. 123–142, Oct. 2018.

[23] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.

[24] J. Xiao, Y. Li, L. Xie, D. Liu, and J. Huang, "A hybrid model based on selective ensemble for energy consumption forecasting in China," *Energy*, vol. 159, pp. 534–546, Sep. 2018.

[25] Y. Wu and J. Gao, "AdaBoost-based long short-term memory ensemble learning approach for financial time series forecasting," *Current Sci.*, vol. 115, no. 1, pp. 159–165, 2018.

[26] L. Xiao, Y. Dong, and Y. Dong, "An improved combination approach based on AdaBoost algorithm for wind speed time series forecasting," *Energy Convers. Manage.*, vol. 160, pp. 273–288, Mar. 2018.

[27] G. E. P. Box and D. A. Pierce, "Distribution of residual autocorrelations in autoregressive-integrated moving average time series models," *J. Amer. Statist. Assoc.*, vol. 65, no. 332, pp. 1509–1526, Apr. 1970.

[28] J. D. Hamilton, *Time Series Analysis*, vol. 2. Princeton, NJ, USA: Princeton Univ. Press, 1994.

[29] H. Qiu, S. Xu, F. Han, H. Liu, and B. Caffo, "Robust estimation of transition matrices in high dimensional heavy-tailed vector autoregressive processes," in *Proc. Int. Conf. Mach. Learn.*, vol. 37, 2015, p. 1843.

[30] I. Melnyk and A. Banerjee, "Estimating structured vector autoregressive models," in *Proc. Int. Conf. Mach. Learn.*, 2016, pp. 830–839.

[31] Y. You, J. Demmel, C.-J. Hsieh, and R. Vuduc. (2018). "Accurate, fast and scalable kernel ridge regression on parallel and distributed systems." [Online]. Available: https://arxiv.org/abs/1805.00569

[32] A. J. Smola and B. Schölkopf, "A tutorial on support vector regression," *Statist. Comput.*, vol. 14, no. 3, pp. 199–222, Aug. 2004.

[33] H. Yang, K. Huang, I. King, and M. R. Lyu, "Localized support vector regression for time series prediction," *Neurocomputing*, vol. 72, nos. 10–12, pp. 2659–2669, 2009.

[34] F. Tobar, I. Castro, J. Silva, and M. Orchard, "Improving battery voltage prediction in an electric bicycle using altitude measurements and kernel adaptive filters," *Pattern Recognit. Lett.*, vol. 105, pp. 200–206, Apr. 2018.

[35] L. Jun and W. Qiu-Li, "Short-term traffic flow online forecasting based on kernel adaptive filter," *J. Meas. Sci. Instrum.*, vol. 9, no. 4, pp. 326–334, 2018.

[36] S. Roberts, M. Osborne, M. Ebden, S. Reece, N. Gibson, and S. Aigrain, "Gaussian processes for time-series modelling," *Philos. Trans. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 371, no. 1984, 2013, Art. no. 20110550.

[37] S. Dasgupta and T. Osogami, "Nonlinear dynamic boltzmann machines for time-series prediction," in *Proc. AAAI*, 2017, pp. 1833–1839.

[38] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. Cottrell. (2017). "A dual-stage attention-based recurrent neural network for time series prediction." [Online]. Available: https://arxiv.org/abs/1704.02971

[39] Y. Bengio, P. Simard, and P. Frasconi, "Learning long-term dependencies with gradient descent is difficult," *IEEE Trans. Neural Netw.*, vol. 5, no. 2, pp. 157–166, Mar. 1994.

[40] F. Wang, Y. Yu, Z. Zhang, J. Li, Z. Zhen, and K. Li, "Wavelet decomposition and convolutional LSTM networks based improved deep learning model for solar irradiance forecasting," *Appl. Sci.*, vol. 8, no. 8, p. 1286, 2018.

[41] Y.-Y. Chang, F.-Y. Sun, Y.-H. Wu, and S.-D. Lin. (2018). "A memory-network based solution for multivariate time-series forecasting." [Online]. Available: https://arxiv.org/abs/1809.02105

[42] M. E. Torres, M. A. Colominas, G. Schlotthauer, and P. Flandrin, "A complete ensemble empirical mode decomposition with adaptive noise," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, May 2011, pp. 4144–4147.

[43] C. Bandt and B. Pompe, "Permutation entropy: A natural complexity measure for time series," *Phys. Rev. Lett.*, vol. 88, no. 17, 2002, Art. no. 174102.

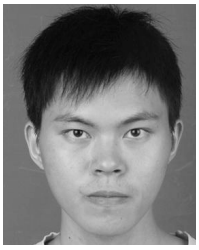
[44] X. Liang *et al.*, "Assessing Beijing's $PM_{2.5}$ pollution: Severity, weather impact, APEC and winter heating," *Proc. Roy. Soc. A, Math., Phys. Eng. Sci.*, vol. 471, no. 2182, 2015, Art. no. 20150257.

[45] T. Hong, P. Pinson, S. Fan, H. Zareipour, A. Troccoli, and R. J. Hyndman, "Probabilistic energy forecasting: Global energy forecasting competition 2014 and beyond," *Int. J. Forecasting*, vol. 32, no. 3, pp. 896–913, 2016.

FAGUI LIU received the M.S. degree from Beihang University, in 1991, and the Ph.D. degree from the South China University of Technology, Guangzhou, China, in 2006, where she is currently a Professor with the School of Computer Science and Engineering. Her research interests include service computing, the Internet of Things, cloud computing, and big data.

MUQING CAI received the B.S. degree from the South China University of Technology, Guangzhou, China, in 2017, where he is currently pursuing the M.S. degree with the School of Computer Science and Engineering. His research interests include machine learning, time series analysis, and the Internet of Things.

LIANGMING WANG is currently a Lecturer with the South China University of Technology, China. His main research interests include network security and the Internet of Things.



YUNSHENG LU received the B.S. degree from the South China University of Technology, Guangzhou, China, in 2018, where he is currently pursuing the M.S. degree with the School of Computer Science and Engineering. His research interests include machine learning, time series analysis, and the Internet of Things.