IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received December 18, 2020, accepted March 2, 2021, date of publication March 15, 2021, date of current version April 20, 2021.

*Digital Object Identifier 10.1109/ACCESS.2021.3065969*

# Multi-Attention Generative Adversarial Network for Multivariate Time Series Prediction

**XIANG YIN[1,2], YANNI HAN[1], HONGYU SUN[3], ZHEN XU[1], HAIBO YU[1], AND XIAOYU DUAN[4]**

[1]Institute of Information Engineering, Chinese Academy of Sciences, Beijing 100093, China
[2]School of Cyber Security, University of Chinese Academy of Sciences, Beijing 100049, China
[3]School of Software and Electrical Engineering, Swinburne University of Technology, Melbourne, VIC 3122, Australia
[4]School of Computer Science, Wuhan University, Wuhan 430072, China

Corresponding author: Yanni Han (hanyanni@iie.ac.cn)

This work was supported in part by the Youth Talent Star of Institute of Information Engineering, Chinese Academy of Sciences under Grant Y7Z0091105, and in part by the National Natural Science Foundation of China under Grant 61771469.

**ABSTRACT** Multivariate Time series data play important roles in our daily life. How to use these data in the process of prediction is a highly attractive study for many researchers. To achieve this goal, in this paper, we present a novel multivariate time series prediction method based on multi-attention generative adversarial network. This method includes three phases to explore multivariate time series prediction. Firstly, the encoder stage consists of two modules, from which the input-attention and self-attention can encode the exogenous sequence into latent space. Secondly, the decoder stage consists of the temporal-convolution-attention module, which can extract long-term temporal patterns. To solve the problem of low accuracy in long-term prediction, inspired by the weight clipping method, we design an improved discrimination network finally. The experiment results indicate that multi-attention mechanism is useful and the discrimination network can improve the performance in multivariate time series prediction. We also tested extensive empirical studies with five real world datasets (NASDAQ100, SML2010, Energy, EEG and Air Quality) demonstrate the effectiveness and robustness of our proposed approach.

**INDEX TERMS** Multivariate data, time series prediction, multi-attention, generative adversarial network.

## I. INTRODUCTION

Multivariate time series data exist in every aspect of our daily life. From the price of the stock market, the traffic flows of highways, the outputs of solar power plants, to the temperatures in different cities, users are often interested in the prediction of new trends or new potential hazardous events based on historical observations on time series signals. The prediction results can provide a basis for various situations as production planning, control, optimization, etc. For example, a better route plan can be devised according to the predicted traffic jam patterns a few hours ago, a larger profit can be gained through the prediction of recent stocks, a reasonable energy forecast can help us carry out load balancing, and a successful temperature forecast can also help us make reasonable travel preparation.

Typically, the periodicity and tend of the data play important roles in predicting future data. People usually focus on short-term or long-term predictions. For now, the autoregressive moving average (ARMA) model [1] is a common time series prediction model. Researchers proposed different improved models based on the ARMA model to solve the different problems, such as the auto-regression integral moving average (ARIMA)model. ARIMA model is a form of regression analysis that measures the strength of a dependent variable relative to other variable variations. However, experiments proved that these models cannot simulate nonlinear relationships. To cope with this problem, A nonlinear autoregressive exogenous (NARX) model propose in this paper. Our model associates the current value of the time series value with the past value of the same sequence and the past value of the driving (exogenous) sequence.

Specifically, real-world applications often entail short-term or long-term patterns. In a set of time steps, various time series components, such as complex trends, seasonality, and noise, can be observed within a certain range. Compared with ARIMA models, ANN (Artificial Neural Network) models are much more complex techniques for training and forecasting. There is significant interest in the Recurrent Neural Network (RNN) and sequence-to-sequence models [4] for

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

forecasting. In the standard RNN structure, there are weights between neurons in the hidden layer. With the continuous progress of the sequence, the front hidden layer will affect the back hidden layer, which leads to the temporal correlation of data that can be captured. However, the loss will also accumulate with the sequence. Then vanishing gradients in the RNN have been addressed in the Long Short-Term Memory (LSTM) [5] and the Gated Recurrent Unit (GRU) [6], by introducing gate-like structures. These LSTM and GRU cell structures are however recurrent and are constant over time. Several studies have shown success with variants of these models [7]–[9].

In the study of time series prediction, some people try to solve the problem by learning and predicting the trend of time series data. Lin *et al.* [10] proposed a hybrid neural network to predict a trend in uniform time series. Besides, in some applications, such as algorithmic trading, it is more achievable to predict the trend of stock price rather than forecast the market absolute values [11]. They are also not good at using the information provided by exogenous(driving) sequences of data. To solve the problem of insufficient application of exogenous sequence information, Qin *et al.* [12] and Liu *et al.* [13] proposed to use encoder and decoder framework to solve the problem, in which encoder can help extract information of exogenous series. In another way, it still does not make good use of the correlation between the driver series and the target series. For example, when we predict the network flow as the target value, we also conclude the relationship between the target value and other driving attributes (such as the click rate, jump rate, page stay time or other attributes) which also contain some information for prediction. In the process of real-time data prediction, the driving series at time T cannot be provided for predicting the target value at the same time T, which is the necessity for the DARNN model. Meanwhile, with the increase of the prediction step, the prediction accuracy cannot be maintained. Zhang *et al.* [14] proposed a generative adversarial network (GAN) structure with MLP as a discriminator and LSTM as a generator to predict financial data. However, these methods are based on the recursive application of single step prediction model for multi-step prediction. If there are prediction errors, such errors will continue to accumulate. In general, we are facing two challenges: the first one is that how to use the correlation between driving series and target well, the second one is that the task of using observed time series in the past to predict the unknown time series in prediction– the larger the predict steps, the harder the problem.

In order to cope with these limitations, we propose a novel model, Multi-Attention based Generative Adversarial Network (MAGAN). Our model is composed of three stages, the encoder network, the generator(decoder) network, and the discriminator network as illustrated in Fig 1. We use the encoder stage to capture the correlation of the multivariate driving time series. The encoder network is composed of input-attention which is extracted the correlation of target data in the current time step and self-attention which is adjusted the proportion of driving series. In the decoder stage, we select the temporal relevance of hidden information with temporal-convolution-attention. In the discrimination stage, convolution layers are used to extract data features and discriminated the generated data with the true data. Considering the short-term situation, a discrimination network does not work well in a short sequence. We just use the encoder network and the decoder network to make up a model named Multi-Attention based Recurrent Neural Networks (MARNN)[15]. The main contributions of this paper are as follows:

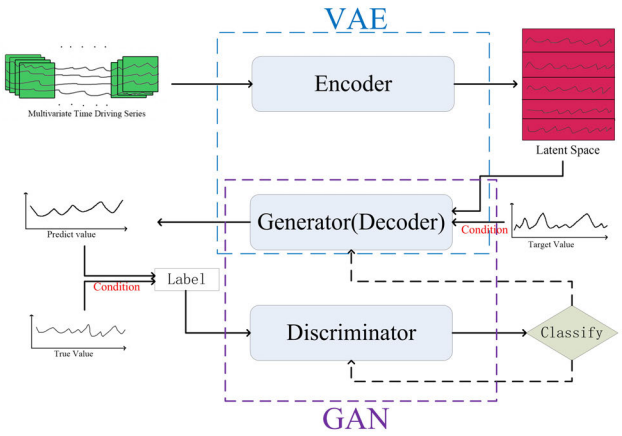



**FIGURE 1.** The architecture of the MAGAN model.

(1) Self-attention dynamically adjusts the proportion of driving series in the encoder network. The higher the weight is, the stronger the correlation of the driving series is. The relevance of the driving series is discussed and proved in section 5.4.3.

(2) We add a convolution layer based on the temporal attention which can capture the temporal pattern features in encoder hidden states. The ability to capture the temporal dependence is enhanced by increasing the weight of the temporal features in the temporal-convolution-attention layer.

(3) We propose a new model MAGAN for long-term prediction. The GAN model can improve the ability of the MARNN model for generating time series data. Results show the proposed framework can efficiently predict the long-term series in future.

(4) We design dynamic weight clipping algorithm, which makes the discriminator stage more stable and accurate. The experimental result proves effective and is better than state-of-the-art methods.

(5) We conduct extensive experimental evaluations on five real datasets, demonstrated the superiority of MAGAN. The effectiveness and robustness of multi attention mechanism and weights clipping are verified.

The remainder of this paper is organized as follows: Section 2 introduces the related background. Section 3 describes the problem statement in the paper. In Section 4, we present the details of our model, including

the Encoder network, Generator network, and Discriminator network. Experiments are given in Section 5. We conclude our work and give a glimpse of the future work in Section 6.

## II. RELATED WORK

The deep learning models for prediction divide into two kinds, one is a discriminative approach which models in view of conditional and another is a generative approach which models in view of joint probability.

### A. DISCRIMINATIVE APPROACHES

Support Vector Machine (SVM) is a representative classical model which is proposed by Vapnik in 1995 [16]. It is a nonlinear time series prediction method based on supervised kernels. Some researchers have proved that the support vector machine method [17] is proposed in order to improve accuracy. It is based on the neural-fuzzy framework to construct different local areas of input space.

Recurrent Structures is another classical network. RNN connects the output and the inputs of neurons so that the network has the ability for remembering trend information. There is a vanishing and exploding gradient problem in standard RNN. Then the LSTM network overcomes this problem by introducing a linear unit that adds the information for each timestep. Therefore, LSTM can maintain temporal information in the state for a long number of timestamps and is widely used in prediction tasks [18] in the field of univariate and multivariate [19]. However, when there are multiple exogenous (driving) sequences available, these networks cannot display the selection of relevant driving series for prediction. In order to address the problem, Qin *et al.* [12] put forward a dual-stage attention-based RNN (DARNN) to capture the long-term temporal dependencies and select the relevant driving series to make the prediction.

CNN (Convolutional Neural Network) is also a common representative deep learning model for forecasting [21]. Some literature used deep CNN layers for dynamic occupancy power grid prediction [22] and wind power prediction [23]. Compared with the widespread use of images, CNN is used to classify rather than forecasting the next value. Until TCN (Temporal Convolution Network) [24] was proposed to provide the ability of CNN for prediction is not worse than RNN.

### B. GENERATIVE APPROACHES

Bayesian Networks (BNs) is a classical model for dealing with uncertainty issues. There have been many studies on time series prediction using the BNs model. Compared with other generative models, BNs can be naturally applied to model the multivariate time series, where the relationship between variables as well as the evolution over time will be both effectively captured [25]–[27].

Deep Belief Network (DBN) is a deep learning model for developing abstract and information abilities. The predicted objects include traffic flow, energy to drought index [28], [29]. But these deep models require much more computational cost.

In recent years, multiple studies have straightforwardly inherited the GAN framework within the temporal setting. The first (C-RNN-GAN) [30] directly applied the GAN architecture to sequential data, using LSTM networks for generator and discriminator. Data is generated recurrently, taking as inputs a noise vector and the data generated from the previous time step. Recurrent Conditional GAN (RCGAN) [31] which is a medical data generation framework took a similar approach, introducing minor architectural differences such as dropping the dependence on the previous output while conditioning on additional input. EEGGAN [32] is a framework for generating brain signals. They improve the training of Wasserstein-GANs to stabilize training and investigate a range of architectural choices critical for time series generation (most notably up and down sampling). Time-series Generative Adversarial Networks [33] is also a data generation approach, which generating realistic time-series data that combines the flexibility of the unsupervised paradigm with the control afforded by supervised training. But all these works are to generate data with the same time trend, not to predict the future data.

In fact, representation learning in the time series primarily deals with the benefits of learning compact encodings for prediction tasks. Meanwhile, in some generation tasks, several works have explored the benefit of combining autoencoders with adversarial training. [34] is proposed for learning similarity measures. [35] is proposed for improving generative capability. But these works are applied to image generation, not data generation.

## III. PRELIMINARY

### A. MULTIVARIATE TIME SERIES STRUCTURE OVERVIEW

Before introducing our proposed model, we give some formal definitions for time series data.

*Definition 1:* A univariate time series $X = [x_1, x_2, \ldots, x_T]$ is an ordered set of real values. The length of X is equal to the number of real values T.

*Definition 2:* An n-dimensional time series $X_t = [x_t^1, x_t^2, \ldots, x_t^n]$ consists of n different univariate time series with $X_t \in R^n, t \in [0, T]$.

*Definition 3:* A dataset $D = \{(X_1, y_1), (X_2, y_2), \ldots, (X_T, y_T)\}$ is a collection of pairs $(X_t, y_t)$.

Where $X_t$ could be multivariate time series with $y_t$ as its corresponding target values. The $X_t$ is always called driving (exogenous) series because of the relationship with the target value. In this paper, we also pay attention to use the correlation between driving series and target values.

### B. PROBLEM STATEMENT

Based on the concept of adversarial training, GAN is basically composed of two competing neural networks, which help them simulate the distribution of any data. GAN has been a great success in the field of image generation since its appearance. In recent years, some authors have put this concept into temporal data generation. But if the prediction data

are generated directly from the random noise Z, the quality of the generated data is not very good. Therefore, we first use the encoder network to process the driving series of data from the original distribution to a normal distribution. So, the result as the latent space contains the information about driving series data. As we can see, $x^k = \left(x_1^k, x_2^k, \ldots, x_T^k\right) \in R^T$ represents the driving series of length $T$ ,and the $x_t = \left(x_t^1, x_t^2, \ldots, x_t^n\right) \in R^n$ denotes a vector of n driving series at time t. We encoder the time series data $X_t$ to calculate the result as latent space Z.

$$Z = E(x_1, x_2, \ldots, x_T) \tag{1}$$

where $E(\cdot)$ is the Encoder network. In the Generator (Decoder) stage, the temporal-attention mechanism is used to automatically select the time steps of the result of the encoder. Given T target values, i.e., $Y = (y_1, y_2, \ldots, y_T) \in R^T$, where T is the length of window size we define. Y denotes all target series during the past T time step. Then the prediction values $\hat{y} = (\hat{y}_{T+1}, \hat{y}_{T+2}, \ldots, \hat{y}_{T+\varepsilon})$ will be calculated with the latent space $Z$ and target series $Y$. $\varepsilon$ represents the prediction time step. Given the previous reading, predict the target series $\hat{y}$.

$$\hat{y} = D(y_1, y_2, \ldots, y_T, Z) \tag{2}$$

where $D(\cdot)$ is the Decoder (Generator) network. We unite the encoder network and decoder network as the model MARNN which could forecast short-term time series. In order to get better long-term prediction results, we use the true values $y = (y_{T+1}, y_{T+2}, \ldots, y_{T+\varepsilon})$ and prediction value $\hat{y}$ to train the discriminator network, and add category labels $L = (L_{T+1}, L_{T+2}, \ldots, L_{T+\varepsilon})$ as conditional variables to guide the discriminator network. Specifically, the discriminator network is trained to minimize the least square loss between its predictions per time step and the labels of the sequence.

$$D_{loss} = LS(y, L) \tag{3}$$

where $LS$ is the least square function. $L$ is a vector of 1s or 0s for series. The generator is trained to 'trick' the discriminator into classifying its outputs as the true data, that is, it wishes to minimize the least square loss between the discriminator's predictions on generated data and the 'true' label, the vector of 1s (we write as 1).

$$G_{loss} = D_{loss}(Z, 1) \tag{4}$$

## IV. THE MULTI-ATTENTION MODEL

In this section, we introduce our proposed model which mainly consists of three modules. Firstly, the encoder network is mainly composed of input-attention modules and self-attention modules. Time series data is encoded by the two modules into latent space. Secondly, the latent space is decoded by the temporal-convolution-attention module to obtain the predicted results. The two modules combine as the MARNN model. Thirdly, the discriminator network classifies data into true data or generated data. The three modules combine as the MAGAN model.

### A. THE ENCODER NETWORK

The encoder network processes the driving series with self-attention and input-attention, calculates the results with LSTM function, and generates the latent space which can maintain the relationship information. Fig.2 shows the details of the encoder network.

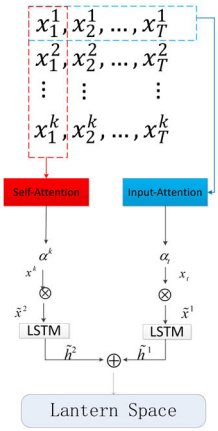



**FIGURE 2.** Details of the encoder network.

#### 1) INPUT ATTENTION

The primary purpose of the attention mechanism is to select from the mass of information that is more critical to the current target. In time series prediction, when long sequence data input into the Encoder-Decoder model, the front information will be covered by the back. Therefore, the important information in the sequence data can be extracted by the attention mechanism to better predict the target value.

Given $T$ input series $x^k = (x_1^k, x_2^k, \ldots, x_T^k) \in R^T$, $T$ is the time window. The attention weight is computed by the following formula.

$$e_t^k = v_e^T \tanh(W_e[h_{t-1}; s_{t-1}] + U_e x^k) \tag{5}$$

$$\alpha_t^k = \frac{\exp(e_t^k)}{\sum_{i=1}^{n} \exp(e_t^i)} \tag{6}$$

where $v_e \in R^T$, $W_e \in R^{T \times 2m}$ and $U_e \in R^{T \times T}$ denote all the learnable parameters. The previously hidden state $h_{t-1}$ and the cell state $s_{t-1}$ are the middle stage results in the encoder LSTM unit. $\alpha_t^k$ is the recording of the attention weight at time $t$. A SoftMax function is used to ensure the attention weights sum to 1. Consider extracting the series adaptively, we multiply the attention weight with the time series data.

$$\tilde{x}^1 = (\alpha_1^k x_1^k, \alpha_2^k x_2^k, \ldots, \alpha_T^k x_T^k) \tag{7}$$

#### 2) SELF-ATTENTION

In order to study textual representation, Vaswani proposed Self-attention [36]. Self-attention is a special case of attention mechanism that only requires a single sequence to compute its representation. The self-attention mechanism achieves great performance to label the deep semantic information [37]. Zheng *et al.* proposed an additive attention

mechanism that dynamically adjusting the proportion of hidden states [38]. We follow a similar idea. In our paper, we note the importance of dynamically adjusting driver sequences so that each driver sequence has a unique adjustment coefficient.

An attention layer with an attention matrix is used to capture the similarity of any token with relative to all adjacent tokens in the input sequence. The input driving series represent as $x_t = (x_t^1, x_t^2, \ldots, x_t^k)$, then the attention mechanism is computed as follows:

$$g_t = \tanh(W_g x_t + b_g) \tag{8}$$

$$\tilde{\alpha}_t^k = \sigma(W_\alpha g_t + b_\alpha) \tag{9}$$

where $\sigma$ denotes the sigmoid function, $W_g \in R^m$ and $W_\alpha \in R^{T\times m}$ are the learnable parameters, $b_g$ and $b_\alpha$ are the bias vectors. The attention weight is then multiplied by the attributes of the driving sequence to show the different importance of the different attributes.

$$\tilde{x}^2 = (\alpha_t^1 x_t^1, \alpha_t^2 x_t^2, \ldots, \alpha_t^k x_t^k)^T \tag{10}$$

We transpose $\tilde{x}^2$ to the same shape with $\tilde{x}^1$ for the next concatenation.

The results are defined as the latent space by calculating with the $f_1$ function.

$$\tilde{h}_t^1 = f_1(\tilde{h}_t^1, \tilde{x}_t^1) \tag{11}$$

$$\tilde{h}_t^2 = f_1(\tilde{h}_t^2, \tilde{x}_t^2) \tag{12}$$

$$Z = [\tilde{h}_t^1; \tilde{h}_t^2] \tag{13}$$

where $f_1$ function is an LSTM unit. $[\tilde{h}_t^1; \tilde{h}_t^2]$ represents the concatenation of the two hidden states. And the output $Z$ put into generator network for next calculating.

### B. THE DECODER NETWORK (GENERATOR NETWORK)

In the encoder(generator) stage, self-attention and input-attention are adopted to better obtain driving series relevance. In the decoder stage, a convolution layer is first used to enhance the temporal learning of the model by applying convolution filters on the row vectors of the encoder latent space Z which is illustrated in Fig.3. The convolutional operations yield $H_i^C \epsilon R^{T\times k}$ is given by:

$$H_i^C = ReLU(\sum_{i=1}^{T} Z \times C) \tag{14}$$

where $k$ filters $C\epsilon R^{1\times w}$ in our model, $1\times w$ represents the size of the kernel. Then temporal attention is employed to adaptively select the hidden state of all time steps related encoder. The attention weight $\beta_t^i$ of each time step $t$ is calculated by the previously hidden state $d_{t-1}$ and the cell state of LSTM unit $s'_{t-1}$.

$$l_t^i = v_d^T \tanh\left(W_d\left[d_{t-1}; s'_{t-1}\right] + U_d H_i^C\right) \tag{15}$$

$$\beta_t^i = \frac{\exp(l_t^i)}{\sum_{j=1}^{T-1}\exp(l_t^j)} \tag{16}$$

where $\left[d_{t-1}; s'_{t-1}\right] \in R^{2p}$ is a concatenation of the previous hidden state and cell state of the LSTM unit. $v_d \in R^m$, $W_d \in R^{m\times m}$,and $U_d \in R^{m\times m}$ are parameters to learn. The attention mechanism computes the context vector $c_t$ with the convolutional operations $H_i^C$ and the attention weights $\beta_t^i$ which represents the importance of the encoder hidden state.

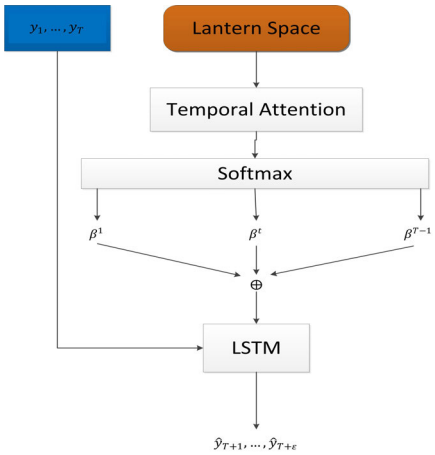



**FIGURE 3.** Details of the decoder (Generator) network.

$$c_t = \sum_{t=1}^{T-1} \beta_t^i H_i^C \tag{17}$$

Then we combine the context vectors $c_{t-1}$ with the given target series $y_{t-1}$ as shown in (18).

$$\tilde{y}_t = \tilde{w}^T [y_{t-1}; c_{t-1}] + \tilde{b} \tag{18}$$

where $[y_{t-1}; c_{t-1}] \in R^{m+1}$ is a concatenation of the target series $y_{t-1}$ and the weighted sum context vectors $c_{t-1}$. $\tilde{w} \in R^{m+1}$ and $\tilde{b} \in R$ are the parameters that map the concatenation to the size of the decoder input. A nonlinear function as an LSTM unit can be used for the update of the decoder hidden state $d_t$ at time $t$ as given in (19).

$$d_t = f_1(d_{t-1}, \tilde{y}_{t-1}) \tag{19}$$

The LSTM unit can capture the time relationship.

### C. THE DISCRIMINATOR NETWORK

The discriminator network is composed of three layers of convolution network. Figure 4 shows the discriminator network illustration. 1D convolution network can better capture the interesting features from overall data and discriminate data.

As we all know, the discriminator is used to judge whether the data is from true data or generator data, and it should give accurate judgment as much as possible. The generator is used to generate data, and the generated data should confuse the discriminator as much as possible. In paper [39], the author thinks that taking cross entropy as a loss will make the generator not optimize the generated data recognized as t data by the discriminator, even if these generated data are still far away from the decision boundary of the discriminator, that is to say, far away from the real data. Why is the least square used as the evaluation of loss function? Because the generator has achieved our goal of confusing the discriminator as much as possible, the cross entropy loss is small. But the least square

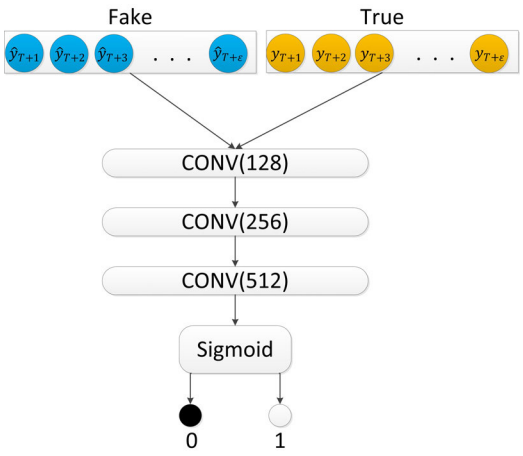



**FIGURE 4.** Details of the discriminator network.

is different. In order to reduce the loss of the least square, the generator must pull the generated data which is far away from the decision boundary to the decision boundary on the premise of confusing the discriminator. So we think that using the least square as the loss function can effectively improve the quality and stability of data generation. The expression of the least squares loss function is as follows:

$$\min_D J(D) = \min_D \frac{1}{2}E_{x\sim p_r}[D(x) - a]^2 + \frac{1}{2}E_{z\sim p_z}[D(G(Z)) - b]^2 \quad (20)$$

$$\min_D J(G) = \min_D \frac{1}{2}E_{z\sim p_z}[D(G(z)) - c]^2 \quad (21)$$

$D(.)$ represent discriminator network, $G(.)$ represent generator network, latent space Z generated by Encoder network. The constants a and b represent the true data and the generated data labels respectively, and c is the value determined by the generator for the discriminator to think the generated data is the true data.

In WGAN [40], a weight clipping method to enforce a Lipschitz constraint is proposed. This method has been proved to have simplicity and already good performance. However, in actual training, all parameters may go to extremes, either maximum or minimum. In this way, the value range of clipping is required to be high. Only when the setting is not large or small, can the generator get a proper return gradient. Therefore, this paper applies a dynamic clipping strategy to solve this problem. First, we get all the parameter values, then calculate the values of the top $\theta$ percent and the last $\theta$ percent of all the parameters, finally use them as the threshold value of the parameters to clip. The MAGAN training process is described in Algorithm 1.

### D. THE TRAINING PROCEDURE

The loss function of my proposed model is mean squared error which can be formulated as:

$$o(\hat{y}_T, y_T) = \frac{1}{N}\sum_{i=1}^{N}(\hat{y}_T^i - y_T^i)^2 \quad (22)$$

**Algorithm 1** MAGAN Training Process

**Require:** $\theta$, the clipping threshold. TrueData, the real world data. Fakedata, the data from generator network. $w$, the weight in discriminator network. 1s, a vector of TrueLabel. 0s, a vector of FakeLabel.

**For** $i$ **in** ganepoch **do**:
  Latern ← MARNN.Encoder
  Fakedata ← MAGAN.Genertator
  **For** $\tau$ **in** Disepoch **do**:
   DlossT ← MAGAN.Discriminator.Train(Truedata,1s)
   DlossF ← MAGAN.Discriminator.Train(Fakedata,0s)
   $w \leftarrow clip(w, w[\theta * len(w)], w[(1-\theta) * len(w)])$
  **End for**
  **For** $\tau$ **in** Genepoch **do**:
   Gloss ← MAGAN.Genertator.Train(1s)
  **End for**
  **For** $\tau$ **in** epoch **do**:
   MARNN.Train()
  **End for**
**End for**

where $\hat{y}_T$ is the predicted vector and $y_T$ is the true vector. Where $N$ is the number of training samples. Adam optimizer [41] and back-propagation algorithm can train our model.

## V. EXPERIMENT AND EVALUATION

### A. DATASET

#### 1) NASDAQ 100 DATA SET (NASDAQ)[42]

The subset of the entire nasdaq100 stock dataset includes 81 major corporations and interpolates the missing data with linear interpolation. The index value of nasdaq100 is used as the target series. These data include 105 days of inventory data from July 26 to December 22 in the 2016 year. Each day contains 390 data points except for 210 data points on November 25 and 180 data points on December 22 which is collected minute-by-minute. In our experience, the last column is the target series and the other 80 columns are the driving series.

#### 2) SML2010 DATA SET (SML) [43]

The dataset is collected from a monitor system mounted in a domestic house. The data were sampled every minute, then the 15-minute average is calculated and uploaded. In our experience, we choose the indoor temperature(room) as the target value and the 18 other features as the driving series.

#### 3) APPLIANCES ENERGY PREDICTION DATA SET(ENERGY) [44]

The dataset is at 10 minutes for about 4.5 months. In our experiment, we employ appliances energy use as the target series, delete the date attribute, and employ other attributes as driving series.

**TABLE 1. The six indexes of time series prediction over the four datasets ($\varepsilon = 1$).**

| Models | | NASDAQ100 | SML2010 | Energy | EEG | AQ |
|---|---|---|---|---|---|---|
| LSTM(1997) | MSE | 0.18159 | 0.07978 | 0.95126 | 0.51442 | 0.24116 |
| | RMSE | 0.42616 | 0.28245 | 0.97532 | 0.71723 | 0.49108 |
| | MAE | 0.35651 | 0.21677 | 0.52727 | 0.41551 | 0.17682 |
| | MAPE(%) | 51.222 | 259.869 | 215.482 | 384.145 | 431.256 |
| | SMAPE | 0.44155 | 0.47647 | 0.92662 | 0.79771 | 0.56843 |
| | R2 | 0.84138 | 0.89488 | 0.52411 | 0.43251 | 0.78995 |
| Seq2Seq(2014) | MSE | 0.60124 | 0.06788 | 1.02599 | 0.45117 | 0.62430 |
| | RMSE | 0.77540 | 0.26054 | 1.01291 | 0.67169 | 0.79013 |
| | MAE | 0.61133 | 0.20334 | 0.54669 | 0.39718 | 0.23847 |
| | MAPE(%) | 45.8007 | 630.050 | 164.344 | 257.622 | 337.681 |
| | SMAPE | 0.59956 | 0.46653 | 1.07271 | 0.68181 | 0.57991 |
| | R2 | 0.47483 | 0.91056 | 0.57197 | 0.54119 | 0.80120 |
| Temporal-attn-RNN(2017) | MSE | 0.63361 | 0.02405 | 0.37958 | 0.31599 | 0.12699 |
| | RMSE | 0.79580 | 0.15508 | 0.61610 | 0.56212 | 0.35636 |
| | MAE | 0.55880 | 0.13108 | 0.18459 | 0.35174 | 0.09509 |
| | MAPE(%) | 40.6236 | 64.0863 | 105.178 | 174.611 | 215.912 |
| | SMAPE | 0.49296 | 0.29459 | 0.54414 | 0.71515 | 0.37451 |
| | R2 | 0.44651 | 0.97300 | 0.67148 | 0.66192 | 0.93228 |
| DARNN(2017) | MSE | 0.00501 | 0.01015 | **0.28175** | 0.18439 | 0.09295 |
| | RMSE | 0.07083 | 0.10076 | **0.53080** | 0.42940 | 0.35698 |
| | MAE | 0.05420 | 0.08149 | 0.19332 | 0.33115 | 0.09295 |
| | MAPE(%) | 6.14165 | 70.5867 | 93.1724 | **90.8817** | 198.115 |
| | SMAPE | 0.0599 | 0.28236 | **0.66817** | 0.60978 | 0.38477 |
| | R2 | 0.99504 | 0.98211 | 0.83145 | 0.71541 | 0.94151 |
| TCN(2018) | MSE | 0.06279 | 0.05316 | 0.31975 | 0.18446 | 0.11742 |
| | RMSE | 0.25058 | 0.23057 | 0.56546 | 0.42949 | 0.34266 |
| | MAE | 0.15976 | 0.18483 | 0.17648 | **0.29493** | 0.09347 |
| | MAPE(%) | 10.6014 | 132.450 | 95.8873 | 92.1225 | 154.714 |
| | SMAPE | 0.11369 | 0.45707 | 0.71775 | **0.57145** | 0.35178 |
| | R2 | 0.94516 | 0.93034 | 0.82671 | 0.74217 | 0.94276 |
| MARNN | MSE | **0.00132** | **0.00148** | 0.28911 | **0.17688** | **0.06938** |
| | RMSE | **0.03634** | **0.03847** | 0.53768 | **0.42058** | **0.26341** |
| | MAE | **0.02344** | **0.02581** | **0.17532** | 0.33353 | **0.05354** |
| | MAPE(%) | **3.73919** | **57.2669** | 97.6681 | 101.951 | **98.1174** |
| | SMAPE | **0.03507** | **0.12889** | 0.67912 | 0.61427 | **0.29341** |
| | R2 | **0.99876** | **0.99727** | **0.86117** | **0.76577** | **0.95447** |

#### 4) EEG STEADY-STATE VISUAL EVOKED POTENTIAL SIGNALS DATA SET(EEG)[45]

This dataset consists of 30 subjects performing Brain Computer Interface for Steady State Visual Evoked Potentials (BCI-SSVEP), and we only use the visual image search dataset from the first subject. In our experiment, we use O1 as the target value and the other 13 signal attributes coming from the electrodes as exogenous series.

#### 5) AIR QUALITY DATA SET (AQ) [46]

The dataset contains 9358 hourly -average response instances from an array of five metal oxide chemical multi-sensors devices. This is the longest free data recorded for the response of air quality chemical sensor devices deployed on site range one year. This dataset has 15 driving series, of which 12 attributes were selected in this paper, and two properties date and time are abandoned. The forecast target value is Temperature.

In our experiments, the last twenty percent points are the test data. Among the rest eighty percent data, the previous eighty percent data points are the training data and the last twenty percent points are the validation data. In order to make each feature make the same contribution to the results, the normalization method is used to preprocess the data.

### *B. BASELINE*

#### 1) LSTM [5]

LSTM is a kind of temporal recurrent neural network, which is specially designed to solve the long-term dependence problem existing in general RNN (Recurrent neural network). All RNN models have a chain form of the recurrent neural network module.

#### 2) SEQ-TO-SEQ [4]

Seq2Seq model is an Encoder-Decoder model. The encoder network can turn a variable length input sequence into a fixed length vector. Then decoder network can decode the vector into a variable length output sequence. This method has good performance in machine translation, text translation, or other NLP processing.

#### 3) DARNN [12]

This algorithm is proposed in 2017, which is a seq2seq model combine with attention mechanism. It shows the state-of-the-art performance in single-step time series prediction.

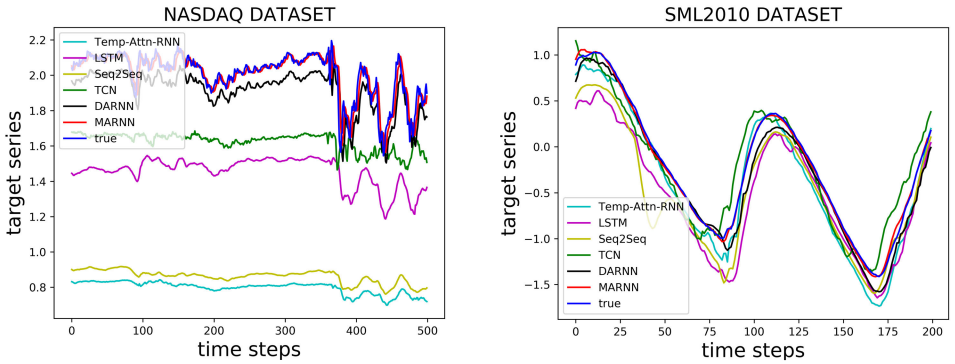



**FIGURE 5.** The detail of time series prediction results over NASDAQ (left) and SML datasets(right).

#### 4) TEMPORAL-ATTENTION-RNN [12]

In the DARNN model, we remove input-attention from the encoder module, while retaining temporal-attention in the decoder module.

#### 5) TCN [23]

The Temporal Convolution Network integrates the modeling ability in the time domain and the feature extraction ability in a low parameter number of convolutions. It runs faster than RNN and is good at capturing temporal dependency.

### C. PARAMETER SETTING AND EVALUATION METRICS

#### 1) HYPER-PARAMETERS

Following the previous work[12], we set seven parameters in our prediction model. During the training phase, the learning rate is 0.001 and the batch size is 128. In our model, we set the length of window size $T$ as the set value of $\{5, 8, 10, 13, 15, 20\}$. When $T$ equals 10 or 13, the prediction result reaches the best performance over the validation set. For simplicity, we use the same hidden dimensionality at the encoder ($m$), the decoder ($p$), and the number of filters ($k$) in convolution, and conduct a grid search over $\{16, 32, 64, 128, 256\}$. When $m = p = k = 64$ or 128, our approach achieves the best performance over the validation set. The size of kernel ($w$) is range as $\{3, 5, 7, 9, 11\}$ in our test and the best performance is reached when $w$ is 7.We test all these approaches including the baseline approaches and record the average performance and standard deviations for comparison.

#### 2) EVALUATION METRICS

In order to compare the effectiveness of various time series prediction algorithms, we use six common criteria to evaluate our model, namely mean squared error (MSE), root mean squared error (RMSE) [47], mean absolute error (MAE), mean absolute percentage error (MAPE), symmetric mean absolute percentage error (SMAPE) and $R^2$ score ($R^2$) which are widely used in regression tasks. The smaller the values of the first five indicators are, the smaller the deviation between the predicted results and the real values, the more accurate the prediction is. The closer the value of the $R^2$ index is to 1, the better the effect is. The formulas of the six measurements are defined below:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N}(y_t^i - \hat{y}_t^i)^2 \tag{23}$$

$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_t^i - \hat{y}_t^i)^2} \tag{24}$$

$$\text{MAE} = \frac{1}{N}\sum_{i=1}^{N}|y_t^i - \hat{y}_t^i| \tag{25}$$

$$\text{MAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_t^i - \hat{y}_t^i|}{y_t^i} \tag{26}$$

$$\text{SMAPE} = \frac{1}{N}\sum_{i=1}^{N}\frac{|y_t^i - \hat{y}_t^i|}{(|y_t^i| + |\hat{y}_t^i|)/2} \tag{27}$$

$$R^2 = 1 - \frac{\sum_{i=1}^{N}(y_t^i - \hat{y}_t^i)^2}{\sum_{i=1}^{N}(y_t^i - \bar{y}_t^i)^2} \tag{28}$$

where $y_t$ is the true target at time t and $\hat{y}_t$ is the predicted value at time $t$.

### D. RESULTS AND DISCUSSION

In this section, we mainly verify the accuracy and robustness of MARNN and MAGAN through experiments. We also interpret our model and prove the efficiency of the multi-attention mechanism.

#### 1) SHORT-TERM PREDICTION

To evaluate the performance of our work, our model is compared with other widely used baseline models in this section. Consider the fairness, the average recorders are displayed in Table 1. In LSTM and Seq2Seq models, we set the LSTM units are same 64. In the DARNN model, the optimal parameters we used are the same as their papers.We use 8 as the depth of the network and 7 as the kernel size in the TCN model. **The results in boldface are the best performance in Table** 1.

As we can observe from Table 1, the algorithms MARNN outperforms other algorithms in six indexes on the five datasets. Among several RNN models, the performance of the RNN with temp-attention has been significantly improved

**TABLE 2.** The six indexes of time series prediction over the four datasets ($\varepsilon = 50$).

| Models | | NASDAQ100 | SML2010 | Energy | EEG | AQ |
|---|---|---|---|---|---|---|
| LSTM(1997) | MSE | 0.3389 | 0.7145 | 0.5968 | 0.4304 | 0.2890 |
| | RMSE | 0.5821 | 0.8452 | 0.7725 | 0.6560 | 0.5376 |
| | MAE | 0.4362 | 0.7924 | 1.3416 | 0.3977 | 0.4184 |
| | MAPE(%) | 178.65 | 468.18 | 251.97 | 195.66 | 346.20 |
| | SMAPE | 0.5142 | 0.8714 | 0.9591 | 0.9754 | 0.6349 |
| | R2 | 0.7311 | 0.4414 | 0.1471 | 0.2143 | 0.3350 |
| Seq2Seq(2014) | MSE | 0.3369 | 0.6451 | 0.6147 | 0.4816 | 0.3582 |
| | RMSE | 0.5804 | 0.8031 | 0.7840 | 0.6939 | 0.5985 |
| | MAE | 0.3491 | 0.7594 | 1.2385 | 0.3814 | 0.4733 |
| | MAPE(%) | 168.28 | 514.71 | 200.17 | 198.26 | 403.42 |
| | SMAPE | 0.4646 | 0.8452 | 0.9257 | 0.8791 | 0.6749 |
| | R2 | 0.7281 | 0.4326 | 0.1879 | 0.2599 | 0.2758 |
| Temporl-attiton-rnn(2017) | MSE | 0.3019 | 0.4617 | 0.6015 | 0.3955 | 0.2537 |
| | RMSE | 0.5494 | 0.6794 | 0.7755 | 0.6288 | 0.5036 |
| | MAE | 0.1826 | 0.5864 | 1.2297 | 0.3714 | 0.3178 |
| | MAPE(%) | 123.51 | 401.82 | 188.65 | 192.12 | 278.11 |
| | SMAPE | 0.3934 | 0.8138 | 0.9318 | 0.8543 | 0.6431 |
| | R2 | 0.7747 | 0.5211 | 0.1871 | 0.2994 | 0.4368 |
| DARNN(2017) | MSE | 0.2881 | 0.3938 | 0.5912 | 0.3817 | 0.2129 |
| | RMSE | 0.5367 | 0.6275 | 0.7688 | 0.6178 | 0.4614 |
| | MAE | 0.1554 | 0.5134 | 1.1561 | 0.2951 | 0.2756 |
| | MAPE(%) | 51.992 | 371.61 | 184.11 | 189.99 | 201.64 |
| | SMAPE | 0.31545 | 0.6841 | 0.8157 | 0.8271 | 0.4712 |
| | R2 | 0.8412 | 0.5442 | 0.2395 | 0.3132 | 0.6143 |
| TCN(2018) | MSE | 0.2412 | 0.4989 | 0.5983 | 0.3591 | 0.1146 |
| | RMSE | 0.4911 | 0.7063 | 0.7734 | 0.5992 | 0.3386 |
| | MAE | 0.0993 | 0.5614 | 1.1154 | 0.2411 | 0.2666 |
| | MAPE(%) | 24.856 | 227.35 | 165.93 | 164.70 | 179.15 |
| | SMAPE | 0.2777 | 0.9310 | 0.8463 | 0.7124 | 0.4052 |
| | R2 | 0.8435 | 0.4161 | 0.2293 | 0.3545 | 0.7661 |
| MARNN | MSE | 0.2581 | 0.3785 | 0.6851 | 0.3792 | 0.1219 |
| | RMSE | 0.5080 | 0.6152 | 0.8277 | 0.6157 | 0.3491 |
| | MAE | 0.1720 | 0.4317 | 1.1941 | 0.2524 | 0.3177 |
| | MAPE(%) | 31.581 | 175.30 | 139.54 | 167.84 | **150.94** |
| | SMAPE | 0.3514 | 0.6871 | 0.8199 | 0.6879 | **0.3814** |
| | R2 | 0.8187 | 0.5307 | 0.3018 | **0.4442** | 0.7526 |
| MAGAN | MSE | **0.1821** | **0.1627** | **0.4135** | 0.3869 | **0.1071** |
| | RMSE | **0.4267** | **0.4033** | **0.6430** | 0.6220 | **0.3272** |
| | MAE | **0.0584** | **0.3233** | **0.8127** | **0.2069** | 0.2847 |
| | MAPE(%) | **18.647** | **121.51** | **131.64** | 164.78 | 164.51 |
| | SMAPE | 0.2783 | **0.6417** | **0.7174** | **0.6635** | 0.4018 |
| | R2 | 0.8249 | **0.5958** | **0.3581** | 0.4399 | 0.7466 |

because the temporal attention mechanism selects relevant encoder hidden states in all time steps to improve the performance. The results of the DARNN model show that the adaptive extraction of the driving series can provide more reliable input features and make more accurate predictions. In our model, the self-attention network is added to the Encoder module to calculate the correlation of the driving series, adjust the weight of the driving series, and improve the prediction accuracy. From the six indicators, we can see that our algorithm has been improved. In order to understand the performance of our model and baseline models in prediction better, we select the data points from timestamp [5500-6000] in the NASDAQ test dataset and timestamp [0-200] in the SML test dataset, and compare our model with other five models. The specific results are shown in Fig. 5.

Observing Fig. 5, we can clearly see that the red curve which represents the prediction data by our proposed model is more consistent with the blue curve which represents the real data. This means that our model has better performance than the other five base models. Meanwhile, we can see that DARNN and MARNN models which have the input-attention in the encoder network outperform the Temp-Attn-RNN model which removes input-attention from the DARNN model. It proves that attention in the encoder network can capture the short dependence.

### 2) LONG-TERM PREDICTION

In this section, we compare our model with the other five baseline models on five data sets. We predict fifty time-steps and record the average of the results in Table 2. In the LSTM

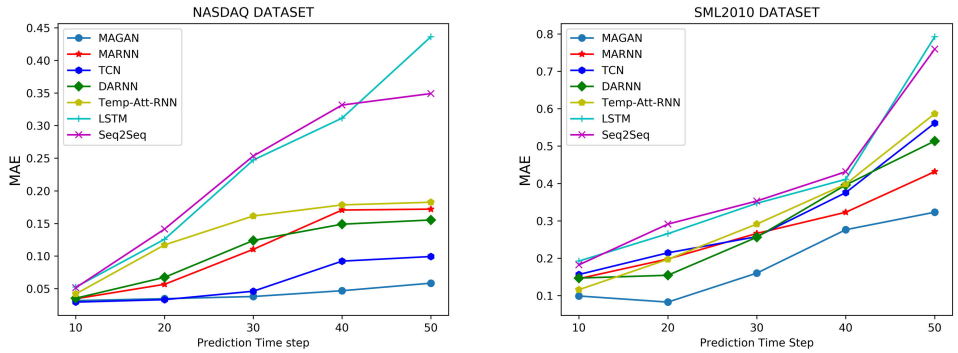



**FIGURE 6.** MAE vs. Length of prediction timesteps ($\varepsilon$) over SML (left) and NASDAQ (right) datasets.

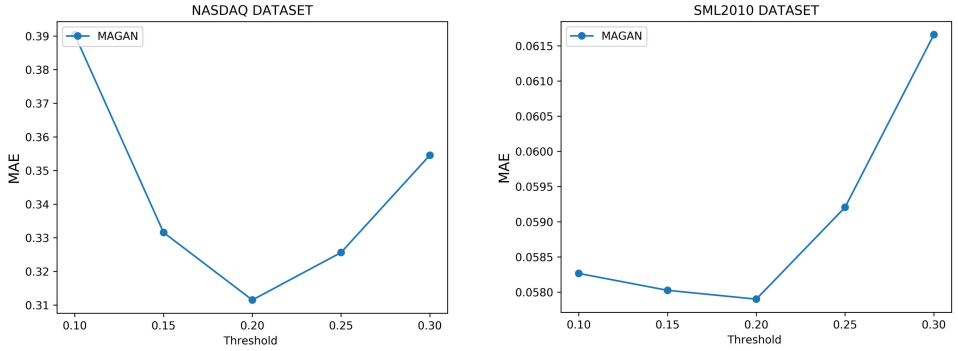



**FIGURE 7.** MAE vs. clipping threshold over SML (left) and NASDAQ (right) Datasets.

and seq2seq models, LSTM units are 64. We use 8 as the depth of the network and 7 as the kernel size in the TCN model. The MARNN model uses the same parameters with MAGAN in this test. **The best result displays in boldface.**

From the results shown in Table 2, we can see that both the LSTM and seq2seq models can maintain temporal dependence, but the advantage of the seq2seq model is that it can output indefinite length values. In the encoder stage, the data will be mapped into a fixed dimension vector, which leads to some information lost. So, the prediction effect is not good enough. In contrast, MARNN is also an encoder-decoder network, but in the encoder stage, the data information is retained to the maximum extent through input attention and self-attention, so the prediction results are better. Although the TCN model can capture temporal dependence, it is not ideal when the prediction length is long, and because of the poor ability to transfer learning, it has a great impact on different databases. We can see from the table that the performance of the MARNN model is a little worse than the TCN model. Meanwhile, based on the MARNN model, the MAGAN model adds a discriminator to train the generated data, which can generate better quality prediction data through feedback more effectively.

Since our algorithm aims at evaluating the effect of multi-step prediction, we explore the variation tendency between the prediction effect and the prediction step size $\varepsilon$ increasing on the SML dataset and the NASDAQ dataset in Figure 6.

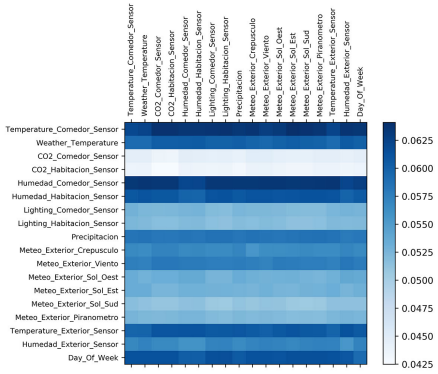



**FIGURE 8.** The heat map of self-attention matrix.

As we can see, with the increase of prediction step size, the prediction accuracy also decreases at different levels. At the same time, the performance of our algorithm is equivalent to the TCN model in short-term prediction. But with the increase of the prediction step, our algorithm performance is more stable and better in long-term prediction finally.

### 3) MODEL INTERPRETATION AND ATTENTION MECHANISM

In order to evaluate the sensitivity and effectiveness of the dynamic weight clipping strategy, we test the effect of different thresholds with the range from 0.1 to 0.3 on the prediction

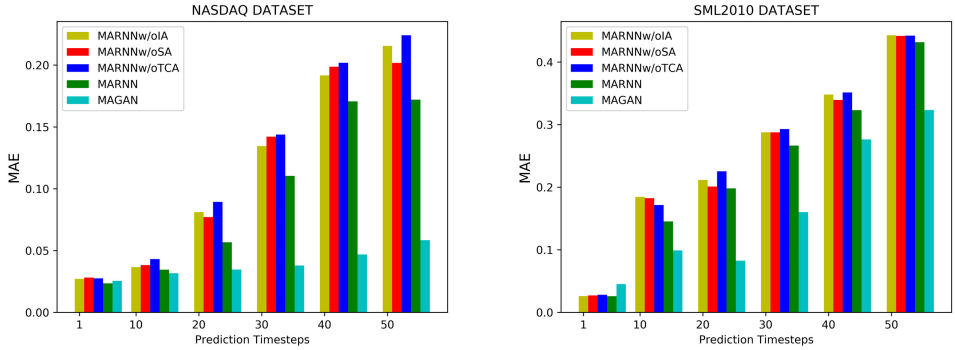



**FIGURE 9.** Results of our model in the ablation tests on the NASDAQ(left) and SML2010(right) datasets.

results. Figure 7 shows the effect of weight clipping. It can be seen the threshold can effectively improve the accuracy of prediction. When the clipping threshold is equal to 0.2, the performance of the model is superior.

To further investigate our method and demonstrate the correlation of driving series, we conducted a case study of self-attention. The NASDAQ dataset has 81 driving series and lock of relevant presentations. Therefore, we use Fig. 8 to show the self-attention result on the SML2010 dataset. We can observe that the ''Tempera-ture_Comedor_Sensor'' attribute which represents ''Relative temperature (dining room)'' and the ''Humedad_Comedor_Sensor'' attribute which represents as ''Relative humidity (dining room)'' contribute a lot to the prediction. Meanwhile, the ''CO2_Comedor_Sensor'' attribute which represents ''Carbon dioxide in ppm (dining room)'' and the ''CO2_Habitacion_Sensor'' attribute which represents ''Carbon dioxide in ppm (room)'' contribute less for prediction. In fact, our target value—Indoor temperature (room) is highly correlated with indoor temperature (dining room) and relative humidity (dining room). Indoor CO2 ppm has little influence on the correlation. This example analysis shows that our method is effective and can be easily interpreted.

To demonstrate the efficiency of the multi-attention mechanism, a careful ablation study is conducted. Specifically, we remove each attention component at a time in our framework and keep them with the same parameters. First, we name the models without different components as follows.

- MARNNw/oIA: The MARNN model without Input-Attention component.
- MARNNw/oSA: The MARNN model without Self-Attention component.
- MARNNw/oTCA: The MARNN model without Temporal-Convolution-Attention component.

Several observations from Fig.9 are worth highlighting. Although, the MAGAN model has not a good performance in short-term prediction. But, with the increasing of the prediction timesteps, the MAGAN model could show its advantages. The MARNN model which has multi-attention always outperforms other ablation models. The MARNNw/oTCN model performs a little worse than the other two ablation models. We think that is because Temporal-Convolution-Attention can enhance the temporal capture ability.

## VI. CONCLUSION

In this work, a Multi-attention model for multivariate time series prediction has been proposed. We use the encoder network to deal with the exogenous sequence of multidimensional data and input the results into the generator as a latent space. Compared with generating data from noise, we can retain more relevant information. At the same time, in order to improve the quality of data generation, a discriminator is used to adjust the data. We also adopt a dynamic threshold method for the discriminator network. Finally, we evaluate with five open real-world datasets. It is proved that our proposed model can achieve better performance compared with five baseline methods in prediction on the six metrics. In the future, based on the GAN framework, we will focus on generating long-term data to solve the duplicate data problem. We also hope to improve the training speed.

## REFERENCES

[1] P. Whittle, *Hypothesis Testing in-Time-Series Analysis*. Uppsala, Sweden: Almqvist and Wiksell, 1951.
[2] D. Asteriou and S. G. Hall, ''ARIMA models and the Box–Jenkins methodology,'' *Appl. Econ.*, vol. 2, no. 2, pp. 265–286, 2011.
[3] T. Lin, B. G. Horne, P. Tino, and C. L. Giles, ''Learning long-term dependencies in NARX recurrent neural networks,'' *IEEE Trans. Neural Netw.*, vol. 7, no. 6, pp. 1329–1338, Nov. 1996.
[4] I. Sutskever, O. Vinyals, and Q. V. Le, ''Sequence to sequence learning with neural networks,'' in *Proc. Adv. Neural Inf. Process. Syst.*, 2014, pp. 3104–3112.
[5] S. Hochreiter and J. Schmidhuber, ''Long short-term memory,'' *Neural Comput.*, vol. 9, no. 8, pp. 1735–1780, 1997.
[6] K. Cho, B. van Merrienboer, C. Gulcehre, D. Bahdanau, F. Bougares, H. Schwenk, and Y. Bengio, ''Learning phrase representations using RNN encoder–decoder for statistical machine translation,'' in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2014, pp. 1724–1734.
[7] L. Zhu and N. Laptev, ''Deep and confident prediction for time series at Uber,'' in *Proc. IEEE Int. Conf. Data Mining Workshops (ICDMW)*, Nov. 2017, pp. 103–110, doi: 10.1109/ICDMW.2017.19.
[8] N. Laptev, J. Yosinski, L. E. Li, and S. Smyl, ''Time-series extreme event forecasting with neural networks at Uber,'' in *Proc. Int. Conf. Mach. Learn.*, vol. 34, 2017, pp. 1–5.

[9] D. C. Maddix, Y. Wang, and A. Smola, "Deep factors with Gaussian processes for forecasting," 2018, *arXiv:1812.00098*. [Online]. Available: http://arxiv.org/abs/1812.00098

[10] T. Lin, T. Guo, and K. Aberer, "Hybrid neural networks for learning the trend in time series," in *Proc. IJCAI*, Aug. 2017, pp. 2273–2279.

[11] Y. Xu and S. B. Cohen, "Stock movement prediction from tweets and historical prices," in *Proc. ACL*, 2018, pp. 1970–1979.

[12] Y. Qin, D. Song, H. Chen, W. Cheng, G. Jiang, and G. W. Cottrell, "A dual-stage attention-based recurrent neural network for time series prediction," in *Proc. IJCAI*, Aug. 2017, pp. 2623–2627.

[13] Y. Liu, C. Gong, L. Yang, and Y. Chen, "DSTP-RNN: A dual-stage two-phase attention-based recurrent neural network for long-term and multivariate time series prediction," *Expert Syst. Appl.*, vol. 143, Apr. 2019, Art. no. 113082.

[14] K. Zhang, G. Zhong, J. Dong, S. Wang, and Y. Wang, "Stock market prediction based on generative adversarial network," *Procedia Comput. Sci.*, vol. 147, pp. 400–406, Jan. 2019, doi: 10.1016/j.procs.2019.01.256.

[15] X. Yin, Y. Han, H. Sun, Z. Xu, H. Yu, and X. Duan, "A multivariate time series prediction schema based on multi-attention in recurrent neural network," in *Proc. ISCC*, Jul. 2020, pp. 1–7.

[16] V. N. Vapnik, *The Nature of Statistical Learning Theory*. Springer, 1995.

[17] A. Miranian and M. Abdollahzade, "Developing a local least-squares support vector machines-based neuro-fuzzy model for nonlinear and chaotic time series prediction," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 24, no. 2, pp. 207–218, Feb. 2013.

[18] Z. C. Lipton, D. C. Kale, C. Elkan, and R. Wetzell, "Learning to diagnose with LSTM recurrent neural networks," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2016.

[19] R. Fu, Z. Zhang, and L. Li, "Using LSTM and GRU neural network methods for traffic flow prediction," in *Proc. 31st Youth Academic Annu. Conf. Chin. Assoc. Autom. (YAC)*, Nov. 2016, pp. 324–328.

[20] Y. LeCun and Y. Bengio, "Convolutional networks for images, speech, and time series," in *The Handbook of Brain Theory and Neural Networks*, vol. 3361, no. 10. Cambridge, MA, USA: MIT Press, 1995, p. 1995.

[21] S. Hoermann, M. Bach, and K. Dietmayer, "Dynamic occupancy grid prediction for urban autonomous driving: A deep learning approach with fully automatic labeling," in *Proc. IEEE Int. Conf. Robot. Automat. (ICRA)*, May 2018, pp. 2056–2063.

[22] H.-Z. Wang, G.-Q. Li, G.-B. Wang, J.-C. Peng, H. Jiang, and Y.-T. Liu, "Deep learning based ensemble approach for probabilistic wind power forecasting," *Appl. Energy*, vol. 188, pp. 56–70, Feb. 2017.

[23] S. Bai, J. Z. Kolter, and V. Koltun, "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling," 2018, *arXiv:1803.01271*. [Online]. Available: http://arxiv.org/abs/1803.01271

[24] Q. Xiao, C. Chaoqin, and Z. Li, "Time series prediction using dynamic Bayesian network," *Optik*, vol. 135, pp. 98–103, Apr. 2017.

[25] M. Naili, M. Bourahla, M. Naili, and A. Tari, "Stability-based dynamic Bayesian network method for dynamic data mining," *Eng. Appl. Artif. Intell.*, vol. 77, pp. 283–310, Jan. 2019.

[26] M. Das and S. K. Ghosh, "Spatio-temporal prediction of meteorological time series data: An approach based on spatial Bayesian network (SpaBN)," in *Proc. Int. Conf. Pattern Recognit. Mach. Intell.* Cham, Switzerland: Springer, 2017, pp. 615–622.

[27] W. Huang, G. Song, H. Hong, and K. Xie, "Deep architecture for traffic flow prediction: Deep belief networks with multitask learning," *IEEE Trans. Intell. Transp. Syst.*, vol. 15, no. 5, pp. 2191–2201, Oct. 2014.

[28] H. Z. Wang, G. B. Wang, G. Q. Li, J. C. Peng, and Y. T. Liu, "Deep belief network based deterministic and probabilistic wind speed forecasting approach," *Appl. Energy*, vol. 182, pp. 80–93, Nov. 2016.

[29] O. Mogren, "C-RNN-GAN: Continuous recurrent neural networks with adversarial training," 2016, *arXiv:1611.09904*. [Online]. Available: http://arxiv.org/abs/1611.09904

[30] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional GANs," 2017, *arXiv:1706.02633*. [Online]. Available: http://arxiv.org/abs/1706.02633

[31] K. G. Hartmann, R. T. Schirrmeister, and T. Ball, "EEG-GAN: Generative adversarial networks for electroencephalograhic (EEG) brain signals," 2018, *arXiv:1806.01875*. [Online]. Available: http://arxiv.org/abs/1806.01875

[32] J. Yoon, D. Jarrett, and M. van der Schaar, "Time-series generative adversarial networks," in *Proc. Neural Inf. Process. Syst. Conf.*, 2019, pp. 1–11.

[33] A. B. L. Larsen, S. K. Sønderby, H. Larochelle, and O. Winther, "Autoencoding beyond pixels using a learned similarity metric," 2015, *arXiv:1512.09300*. [Online]. Available: http://arxiv.org/abs/1512.09300

[34] A. Makhzani, J. Shlens, N. Jaitly, I. Goodfellow, and B. Frey, "Adversarial autoencoders," 2015, *arXiv:1511.05644*. [Online]. Available: http://arxiv.org/abs/1511.05644

[35] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. NIPS*, 2017, pp. 5997–6008.

[36] Z. Tan, M. Wang, J. Xie, Y. Chen, and X. Shi, "Deep semantic role labeling with self-attention," in *Proc. AAAI*, 2018, pp. 1–8.

[37] G. Zheng, S. Mukherjee, X. L. Dong, and F. Li, "OpenTag: Open attribute value extraction from product profiles," in *Proc. KDD*, Jul. 2018, pp. 1049–1058.

[38] X. Mao, Q. Li, H. Xie, R. Y. K. Lau, Z. Wang, and S. P. Smolley, "Least squares generative adversarial networks," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2813–2821.

[39] M. Arjovsky, S. Chintala, and L. Bottou, "Wasserstein GAN," Jan. 2017, *arXiv:1701.07875*. [Online]. Available: https://arxiv.org/abs/1701.07875

[40] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," in *Proc. Int. Conf. Learn. Represent. (ICLR)*, 2015.

[41] *NASDAQ100 Dataset*. Accessed: 2017. [Online]. Available: https://cseweb.ucsd.edu/~yaq007/NASDAQ100_stock_data.html

[42] *SML2010*. Accessed: 2014. [Online]. Available: http://archive.ics.uci.edu/ml/datasets/SML2010

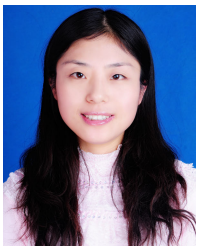
[43] L. M. Candanedo, V. Feldheim, and D. Deramaix, "Data driven prediction models of energy use of appliances in a low-energy house," *Energy Buildings*, vol. 140, pp. 81–97, Apr. 2017.

[44] *EEGDataset*. Accessed: 2018. [Online]. Available: https://archiveics.uci.edu/ml/datasets/EEG+Steady-State+Visual+Evoked+Potential+Signals

[45] *Gas Sensor Array Temperature Modulation Dataset*. Accessed: 2019. [Online]. Available: https://archive.ics.uci.edu/ml/datasets/Gas+sensor+array+temperature+modulation

[46] M. Plutowski, G. Cottrell, and H. White, "Experience with selecting exemplars from clean data," *Neural Netw.*, vol. 9, no. 2, pp. 273–294, Mar. 1996.


**XIANG YIN** was born in Henan, China, in 1990. He received the master's degree from Henan University, in 2018. He is currently pursuing the Ph.D. degree with the Institute of Information Engineering, Chinese Academy of Sciences. His current interests include data mining and time series prediction.



**YANNI HAN** is currently an Associate Professor with the Institute of Information Engineering, Chinese Academy of Sciences. Her current interests include big data intelligent analysis and network security.

**HONGYU SUN** was born in Henan, China, in 1989. He received the master's degree from Lanzhou University, in 2016. He is currently pursuing the Ph.D. degree with the Swinburne University of Technology. His current interests include data mining and anomaly detection.

**HAIBO YU** is currently a Professorate Senior Engineer with the Institute of Information Engineering, Chinese Academy of Sciences. His current interests include international things and security.

**ZHEN XU** is currently a Professorate Senior Engineer with the Institute of Information Engineering, Chinese Academy of Sciences. His current interests include network security and cloud computing.

**XIAOYU DUAN** was born in Henan, China, in 1992. She received the master's degree from Henan University, in 2018. She is currently pursuing the Ph.D. degree with Wuhan University. Her current interests include data mining and anomaly detection.