International Journal for Housing Science and Its Applications
Publish August 10, 2025. Volume 46, Issue 4 Pages 736-747

https://doi.org/10.70517/ijhsa46467

Research on feature extraction and reconstruction method of time series data based on generative adversarial network

Zhi Li[1,*], Yunxia Yin[1] and Yan Yang[1]
[1] The School of Medical Information Engineering, Anhui University of Chinese Medicine, Hefei, Anhui, 230012, China
Corresponding authors: (e-mail: LeeAUCM@163.com).

Abstract Traditional time series analysis methods often face problems such as missing data and noise interference, which affects the predictive ability and accuracy of the model. Generative Adversarial Networks (GANs) have made significant progress in the fields of image generation and data restoration due to their excellent generative capabilities. In this study, a feature extraction and reconstruction method based on generative adversarial network (GAN) for time series data is proposed. By improving the structure of generative adversarial network and introducing the synergistic loss function of global optimization objective and single-channel optimization objective, combined with the Transformer architecture, the CTTS-GAN model is proposed. The model can better retain the features of the original data and improve the diversity of the data when generating time series data. The experimental results show that the data generated by CTTS-GAN exhibits a lower error (0.42) under the maximum mean difference (MMD) metric, showing a distribution closer to the real data. In addition, CTTS-GAN obtained better classification results than the traditional GAN model when using the generated data for the classification task, with a TSTR score of 0.83 on the Support Vector Machine classifier, which is significantly higher than other methods. This indicates that CTTS-GAN has a strong potential for application in generating high quality time series data.

Index Terms Generative Adversarial Networks, Time Series, Feature Extraction, Reconstruction, Transformer, MMD

I. Introduction

With the progress of the information age and the rapid development of science and technology, more and more data are generated in actual production and life, and time series data is a class of ordered, time-stamped sequences of data points [1]. Due to the explosive growth of data, how to accurately identify valuable patterns from data is a difficult task. Several studies have shown that by performing feature extraction on time series data to generate a new form of data, this new data can well reflect the changing trends and morphological features of the original data, and the efficiency of data mining can be improved by using the extracted features for data analysis [2]-[4].

In addition, the feature representation of time series data is to convert the original time series data into data in another domain, which can effectively extract data features and can play the role of data dimensionality reduction [5], [6]. At the same time, feature extraction can make the data in low-dimensional space reflect the important information of the original time series as much as possible [7]. However, for the massive and high-dimensional characteristics of time series data, it is inefficient if tasks such as classification, clustering and prediction are performed directly on the original series data, and algorithms such as Transformer have relatively high time complexity and space complexity [8], [9]. The traditional feature representation of time series data cannot extract the important data features of the time series and is defective in time-frequency analysis. Feature extraction with convolutional networks is limited in multi-scale cycles, and long short-term memory networks have difficulty in modeling long time series dependencies [10]-[12].

In contrast, generative adversarial networks have advanced the theory and practicalization of generative modeling through adversarial training methods. It has a unique and innovative working mechanism, works with generator and discriminator, performs well on periodic feature separation and decoupling of hidden features, and can augment the data to optimize the data quality for better feature extraction [13]-[15].

In this paper, an improved generative adversarial network model, CTTS-GAN, is proposed, which not only integrates traditional convolutional neural networks (CNNs) with generative adversarial networks, but also further enhances the capability of generative modeling through the introduction of the Transformer architecture. During the model training process, the study employs a synergistic loss function with global and single-channel optimization objectives, which enables the generator to ensure the overall structure of the multivariate time series while preserving the key features of the univariate series. In addition, the global-local fusion module (GLFM) proposed in

this paper effectively combines the convolutional self-attention and multi-head self-attention mechanisms to further enhance the quality and diversity of the generated data.

II. Time series generation model

II. A. Time series data

If the values of the sequence data change over time and have a definite value at each definite moment, then these sequence data can be described by the following equation:

$$x = (x_1, x_2, ..., x_n) \subseteq \mathrm{R}^{n \times d} \tag{1}$$

where x_i denotes the value acquired at the moment corresponding to t_i, n is the total number of moments at which the data were captured, and d is the dimension of the data at each moment. When $d = 1$, the data is univariate time series data, and when $d > 1$, the data is multivariate time series data.

When completing the missing multivariate time series data, it is often necessary to set the corresponding mask matrix accordingly, and the corresponding formula of the mask matrix is described as follows:

$$m = (m_1, m_2, ..., m_n) \subseteq \mathrm{R}^{n \times d} \tag{2}$$

Missing locations that do not exist in the multivariate time series data are marked as 1 in the mask matrix accordingly and 0 otherwise.

The fragility of time series data collection devices results in frequent missing data being collected. Thus, the existing data needs to be analyzed in order to complement the missing data [16]. If the method of deleting samples containing missing data is used to construct complete data, there is a situation where too much data is missing and thus cannot be analyzed after deletion. Time-series data may have a large number of missing and drastic changes, which is the main technical difficulty in completing the data.

II. B. Generating Adversarial Networks

Since its invention, the best and most popular task of neural networks is to recognize and classify given things. However, the process of neural network recognition is by generating labels, and then the loss function determines the difference between the generated labels and the actual labels, and then updates the gradient to make the generated labels closer to the actual labels. If you want to use neural networks for creation, you need to determine the large amount of content generated by neural networks, which naturally introduces the idea of generative adversarial networks. Adversarial training is carried out by two neural networks, one neural network generates content similar to the training set, known as the generator, and the other neural network determines whether the content generated by the generator is similar to the training set, known as the discriminator [17]. The basic structure of the generative adversarial network is shown in Fig. 1.

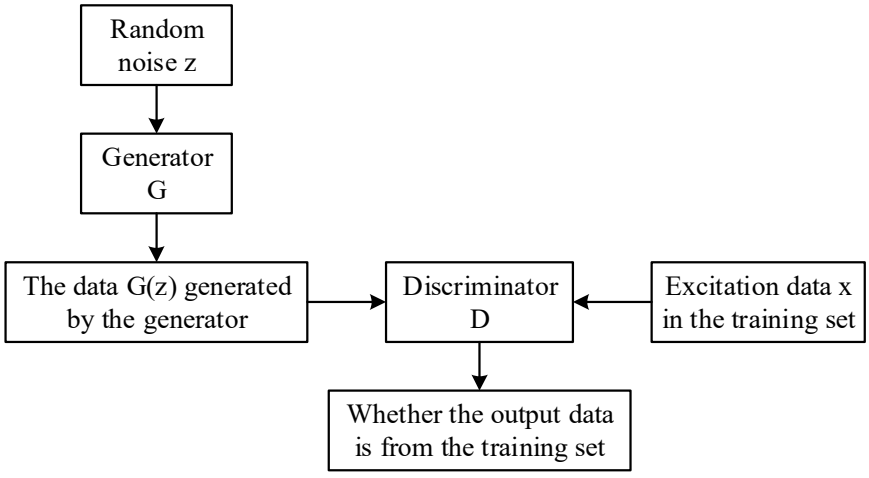


Figure 1: The basic structure of the generative adversarial network

The goal of generative adversarial network training is to obtain generators that are capable of generating the data in the training set, in order to achieve this goal, it is necessary for the generator and the discriminator to play a

dynamic game until both sides reach equilibrium at the end. The loss function of the GAN is described in the following equation:

$$\min_G \max_D V(D,G) = E_{x\sim p_{data}(x)}[\log D(X)] + E_{z\sim p_z(z)}[\log(1-D(G(z)))] \tag{3}$$

When GAN emerged, there was no popular network architecture for both generators and discriminators. Deep Convolutional Generative Adversarial Networks (DCGAN) solves this problem by combining Convolutional Neural Networks (CCNN) with GAN.

The generators and discriminators in DCGAN use modified CNNs, including the replacement of the pooling layer with a convolutional layer, which allows the network to learn how to downsample on its own instead of discarding pixels at regular intervals, as was previously the case. There is also the removal of the fully connected layer from the network as well as the adoption of a bulk regularization (BN) layer. For the generator in DCGAN, the activation function of all layers is ReLU function except for the last layer of the generator, which has tanh; and for the discriminator, the activation function of each layer is LeakyReLU function. The experiment proves that DCGAN improves the stability of GAN training as well as the quality of GAN generated data.

II. C.Sequence-to-sequence neural networks

Sequence-to-sequence neural networks are often used in tasks where both input and output are sequential data and the length of the output cannot be determined in advance. An example is the translation of languages into each other. A sequence-to-sequence model usually consists of an encoder and a decoder. The encoder compresses the sequence data into fixed length vectors, and the decoder recovers the vectors into sequence data.

At each moment, the encoder produces different outputs and states depending on the inputs and the previous moment's state. For the encoder, the state at each moment contains useful information about the current and previous moments, thus discarding the outputs and passing only the state. After processing the input completely, the state, which contains all the input information, is passed to the decoder. Assuming that the encoder's parameters involved in the loop computation are W_E, the parameters involved in the input computation are W_{EI}, and the activation function is f, the encoder's state update can be described by the following equation:

$$h_t = f(W_E h_{i-1} + W_{EI} x_t) \tag{4}$$

The initial state of the decoder is the final state of the encoder, and the initial input of the decoder is fixed to the “START” flag. Assuming that the parameters of the decoder are W_D, the process of updating the state of the decoder is described by the following equation:

$$h_t = f(W_D h_{i-1}) \tag{5}$$

Meanwhile, the decoder produces the output $\hat{y}_t$ at each moment and the gradient is updated by the loss function according to the difference between $\hat{y}_t$ and y_t. Assuming that the parameter that produces the output is W_{DO}, and that the activation function is g, the process of producing the output is described by the equation below:

$$\hat{y}_t = g(W_{DO} h_t) \tag{6}$$

The problem with sequence-to-sequence networks is that even with LSTM or GRU as encoder and decoder, the problem of gradient vanishing still occurs as the length of the sequence increases. In addition to this, when the decoder decodes, part of the output does not require the encoding of the entire input sequence, the output at a certain moment may only be relevant to the input at certain moments, whereas the encoder must encode all the information of the entire sentence, which leads to the neural network having to selectively discard information. The attention mechanism, on the other hand, solves this problem by making the decoder focus only on the information it needs when decoding.

After adding the attention mechanism, the encoder needs to input the hidden state of each moment into the attention layer. When decoding, the decoder needs to calculate the weight value $a_t(s)$ of the current moment's hidden layer state h_t to the encoder's hidden layer state $\bar{h}_s$ at each moment, which is described by the following formula:

$$a_t(s) = \frac{score(h_t, \bar{h}_s)}{\sum_{s'} exp(score(h_t, \bar{h}_{s'}))} \tag{7}$$

score denotes the attention score of the current state with respect to other hidden layer states and is calculated as follows:

$$score(h_t,\bar{h}_s)=\begin{cases}h_t^T\bar{h}_s & Dot\\ h_t^T W_a\bar{h}_s & Genaral\\ v_a^T\tanh(W_a\cdot concat(h_t,\bar{h}_s)) & Concat\end{cases} \tag{8}$$

After completing the computation of the weight scores, the vector c_t that the decoder needs to pay attention to when decoding can be computed as follows:

$$c_t=\sum_s a_t(s)\bar{h}_s \tag{9}$$

With the emergence of Transformer, which relies only on attention mechanisms and feedforward neural networks and has outperformed RNNs in most areas of processing sequence data, the research on Transformer is getting deeper and deeper. Transformer takes the input sequence data and its corresponding position and represents them as vectors, and then sums these vectors together to get the corresponding vector representation of the input. Transformer's encoder is a vectorized representation of the input sequence data and its corresponding position.

The input of Transformer's encoder passes through the multi-head self-attention layer, and then passes through the Add and Norm layer, which is formulated as follows:

$$Output=LayerNorm(Input+MultiHead(Input)) \tag{10}$$

This layer serves as a residual connection to prevent the problem of gradient vanishing when the network is too deep. After this layer, the input is handed off to a two-layer feed-forward neural network with an activation function of ReLU in the first layer and no activation function in the second layer. Assuming that the input to this network is X, the weight matrix of the first layer is W_1 with a bias of b_1, and the weight of the second layer is W_2 with a bias of b_2, the formulation of the layer is described as follows:

$$Output=\max(0,XW_1+b_1)W_2+b_2 \tag{11}$$

III. Improvements in generating adversarial networks

To address the limitations of the loss function in DCGAN and the GAN under the Transformer architecture for the multivariate time series generation task, this chapter proposes a loss function with the synergy of global and single-channel optimization objectives, as well as a GAN structure that incorporates convolution and Transformer improvements.

III. A. Improvement of LSGAN loss function

In order to generate more realistic and higher quality samples, the GAN model needs to focus on both the joint distribution of multivariate sequence data and the marginal distribution of univariate sequences. Therefore, based on the loss function of LSGAN [18], this paper sets the generator and discriminator with a global optimization objective and a single-channel optimization objective, respectively. Through the synergy of the generator and the discriminator, the overall timing structure of the multivariate sequence can be maintained while ensuring that the key features of the univariate sequence are preserved.

Global optimization objective: the discriminator evaluates the differences between the synthetic multivariate time series samples and the real time series samples as a whole, in order to distinguish the real sequence x from the synthetic sequence $G(z)$ as accurately as possible, with the optimization objective:

$$Loss(D_0)=\frac{1}{2}\min_{D_0}(E_{x\sim P_{dima}}[(D_0(x)-a)^2]+E_{z\sim p_z(z)}[(D_0(G(z))-b)^2]) \tag{12}$$

where $p_z(z)$ denotes the distribution that the noise z obeys, p_{data} denotes the joint distribution of the true sequences, a is the labeled value of the true sequences, b is the labeled value of the synthetic sequences, and E denotes the expected value. D_0 denotes the global discriminator.

The goal of the generator is to make the joint distribution of the real and synthetic sequences as similar as possible, which can be expressed as:

$$Loss(G)=\frac{1}{2}\min_{G}E_{z\sim p_z(z)}[(D_0(G(z))-a)^2] \tag{13}$$

Single-channel optimization objective: during the training of the GAN, the input multivariate time series is analogous to a picture whose length is the length of the sequence, width is 1, and the number of channels is the number of variables. For each channel, the discriminator's goal is to distinguish the real sequence from the synthetic sequence as accurately as possible on that channel:

$$Loss(D_i)=\frac{1}{2}\min_{D_i}(E_{x^i\sim p^i_{dam}}[(D_i(x^i)-a)^2]+E_{z\sim p_z(z)}[(D_i(G(z)^i)-b)^2]) \tag{14}$$

where p^i_{data} denotes the distribution of the i th variable of the true sequence. D_i denotes the single-channel discriminator, $G(z)^i$ denotes the i th variable of the generator's synthesized sequence, x^i denotes the i th variable of the true sequence, and $i\in(1,2\cdots d)$.

The goal of the generator is to make the synthesized sequence on each channel as similar as possible to the corresponding real sequence, which can be expressed as:

$$Loss(G)=\frac{1}{2}\min_{G}E_{z\sim p_z(z)}[(D_i(G(z)^i)-a)^2] \tag{15}$$

Therefore, the proposed model in this paper, the loss function of the generator is shown in Eq. (16) and the loss function of the discriminator is shown in Eq. (19), which includes the loss function of the global discriminator in Eq. (17) and the loss function of the single channel in Eq. (18). In standard GAN, the training of generator and discriminator is often unstable, in order to improve the stability of training, this paper adopts label smoothing to correct the value of loss function, which makes the generator and discriminator more balanced in the training process:

$$\begin{aligned}Loss(G)=&\frac{1}{2}\min_{G}(1-\lambda)E_{z\sim p_z(z)}[(D_0(G(z))-(a-\varepsilon))^2]\\&+\frac{1}{2}\min_{G}\lambda\sum_{i=1}^{d}\frac{1}{d}E_{z\sim p_z(z)}[(D_i(G(z)^i)-(a-\varepsilon))^2]\end{aligned} \tag{16}$$

$$\begin{aligned}Loss(D_0)=&\frac{1}{2}\min_{D_0}(E_{x\sim Ba\tan}[(D_0(x)-(a-\varepsilon))^2]\\&+E_{z\sim p_z(z)}[(D_0(G(z))-(b-\varepsilon))^2])\end{aligned} \tag{17}$$

$$\begin{aligned}Loss(D_i)=&\frac{1}{2}\min_{D_i}(E_{x^i\sim dana}[(D_i(x^i)-(a-\varepsilon))^2]\\&+E_{z\sim p_z(z)}[(D_i(G(z)^i)-(b-\varepsilon))^2])\end{aligned} \tag{18}$$

$$LossD=(1-\lambda)Loss(D_0)+\lambda\sum_{i=1}^{d}\frac{1}{d}Loss(D_i) \tag{19}$$

In Eqs. (17)~(19), a and b denote the labeled values of the real sequence and the labeled values of the synthetic sequence during the training process, respectively, and ε denotes the parameter of label smoothing, with $b=0,a=1,\varepsilon=0.1$.

III. B. Improvement of GAN model under Transformer architecture

The generator of TTS-GAN adopts the standard Transformer encoder structure, which consists of multiple encoding blocks stacked together, and each encoding block contains a multi-head attention mechanism and a feed-forward neural network.The structure of the TTS-GAN model is shown in Fig. 2.

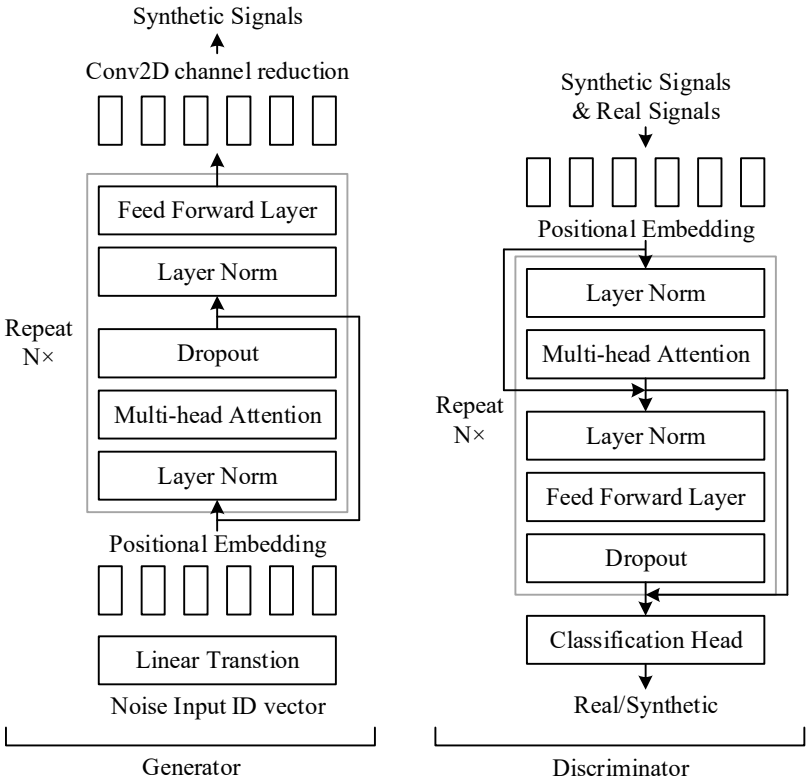


Figure 2: TTS-GAN model structure

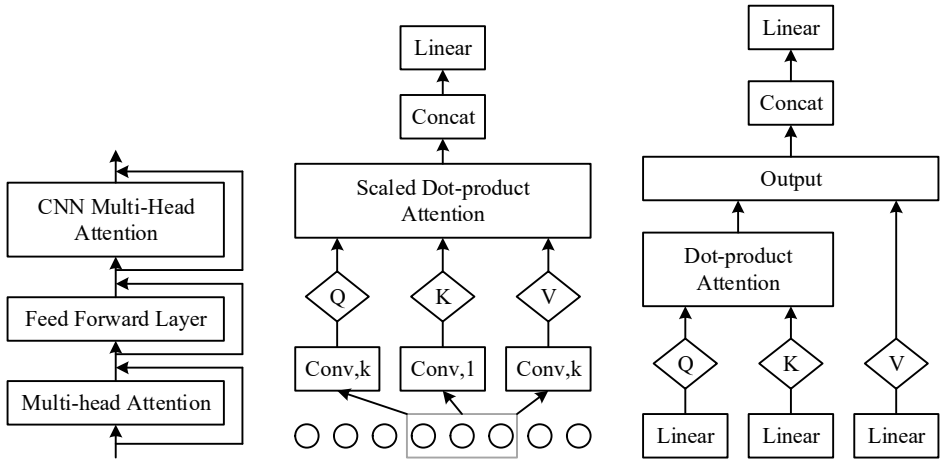
In order to generate higher quality time series, this paper uses the encoder structure of Transformer to construct the generator and discriminator of GAN, different from TTS-GAN, the model CTTS-GAN in this paper designs the global-local fusion module (GLFM) in the generator and uses the loss function composed of global optimization objective and single-channel optimization objective, which is reflected in the model structure of It is reflected in the model structure that not only the global discriminator but also the single-channel discriminator is used.

(1) Global-local fusion module

The global-local fusion module is shown in Fig. 3. (a)~(c) represent the structure of the GLFM module, the convolutional attention computation in GLFM, and the multi-head attention computation in GLFM, respectively.

The GLFM module fuses the convolutional self-attention layer with the multi-head self-attention layer. The results of the multi-head attention layer and the convolutional self-attention are fused by using the residual connection, which enables the generator to capture the features of the time series more comprehensively and accurately.

The computational flow of the multi-head self-attention layer is shown in Fig. 3 (b). In the multi-head self-attention layer, the linear layer maps the input Y into s mutually different query matrices $Q_i = YW_i^Q$, key matrices $K_i = YW_i^K$ and value matrices $V_i = YW_i^v, i \in (1,2\cdots s)$. Here $W_i^Q \in \square^{P_q \times d_q}, W_i^K \in \square^{P_k \times d_k}, W_i^V \in \square^{P_k \times d_1}$ are learnable parameters, p_q, p_k, p_r and d_q, d_k, d_v denote, respectively, W_i^Q, W_i^K, W_i^V in two dimensions.



(a) GLFM module structure

(b) Convolutional attention calculation method in GLFM

(c) Calculation method of multi-head attention in GLFM

Figure 3: Global-Local Fusion Module

Firstly, $Q_i K_i^T$ of the query matrix Q_i and the key matrix K_i are computed to obtain the similarity, which is subsequently obtained by dividing by $\sqrt{d_k}$ and normalizing it by the $Softmax$ function to obtain the score matrix A_i:

$$A_i = Softmax(Q_i K_i^T / \sqrt{d_k}) \tag{20}$$

Then, the score matrix A_i is multiplied with the value matrix V_i to compute the attention matrix Z_i:

$$Z_i = Attention(Q_i, K_i, V_i) = AV_i = Softmax(Q_i K_i^r / \sqrt{d_k}) \cdot V_i \tag{21}$$

The output of the attention matrix $Z_i, (i = 1, 2 \cdots s)$ is linearly transformed to obtain the output of the layer as in Eq. (22), where W_0 is the learnable network weight parameter:

$$Z = Concat(Z_1, Z_2, \cdots, Z_s) W_0 \tag{22}$$

The computational flow of the convolutional self-attention layer is shown in Figure 3 (c), where $conv,k$ denotes a one-dimensional convolution with convolutional kernel size k. The convolutional self-attention layer divides the input X from the previous layer into multiple local blocks $X = \{X_1, X_2, \cdots X_s, \cdots\}$, and then learns Q, K, V for each local block X, using one-dimensional convolutional operations. Among them:

$$Q_{ij}, K_{ij}, V_{ij} = \sum_{p,q} K_{p,q} y_{i+p-\lfloor k/2 \rfloor, j+q-\lfloor k/2 \rfloor} \tag{23}$$

where $K_{p.q} \in R^{C_{sw} \times k \times k}$ denotes the weight of the convolution kernel (p, q) position, k denotes the size of the convolution kernel, C_{in}, C_{out} denotes the size of the channels of the inputs and outputs, and y_i denotes the size of the local feature tensor at feature location (i, j) on the block.

First, the dot product of Q and K is calculated, then divided by the square root of the key vector K dimension $\sqrt{d_k}$, normalized by the $Softmax$ function and multiplied by the value matrix V to obtain the attention matrix Z_0 between blocks and blocks:

$$Z_0 = Softmax(QK^T / \sqrt{d_k}) \cdot V \tag{24}$$

The convolutional self-attention layer performs a convolution operation on the input sequence using h sets of mutually exclusive convolutional kernels and computes their attention matrix $Z_i, i \in (1, 2, \ldots h)$. The final output Z of the layer is obtained by linearly transforming Z_i:

$$Z = Concat(Z_1, Z_2, \cdots Z_h)W \tag{25}$$

III. C. Overall framework of the model

The generator of the CTTS-GAN model proposed in this paper consists of the GLFM module, Layer Normalization (LN) layer, Droupout layer and feedforward neural network layer. The input noise z is first dimensionally transformed and positional coding information is added through the linear and positional embedding layers, the covariate bias within the network is reduced through the LN layer, then the feature representation of the time series is learned step-by-step by cyclic computation through the GLFM module and the MLP module, the overfitting of the network is reduced through the Droupout layer, and finally the information of the modules is fused together using residual linkage and the convolutional layer to adjust the output shape to be the same as the real sample shape.

During training, the input size of the global discriminator is $[Batch\_size, channels, 1, sequence]$ and the input size of the single-channel discriminator is $[Batch\_size, 1, 1, sequence]$. where $Batch\_size$ denotes the number of samples used to update the model parameters in each iteration of training, $channels$ denotes the number of univariate time series contained in the input multivariate sequence, and $sequence$ denotes the length of the input sequence. The input is first divided into multiple blocks through a positional embedding layer, each block is spread, and a soft positional encoding value is added after each spread block to preserve the sequential information of the input. After that, the model learns the feature representation of the input sequence through the cyclic action of the multi-head attention layer and the feedforward layer, and finally obtains the result of the model classification.

IV. Simulation experiments and analysis of results

IV. A. Introduction to the experimental data set

The source of the original real dataset for generating the model input in this chapter is “wind” database, which is the leading financial database in China, and the data content of which includes stocks, funds, bonds, insurance and other fields. In this chapter, the Shanghai Stock Composite Index (stock code 000001) of the stock module is selected as the experimental data set. The value of this sequence of data changes with time and has a definite value at every definite moment, so this paper can choose it as a data source for experimentation.

The time span of the selected SSE Composite Index data set in this experiment is from March 2011 to March 2024, in which the data parameters involved in the training and learning of the generative model are the opening price, the highest price, the lowest price, the closing price and the turnover volume totaling five parameters.

IV. B. Detection and analysis of generated data

In this section, the maximum mean difference (MMD) is used to evaluate how well the generated data fits the real data [19]. An evaluation metrics algorithm based on true training and false testing (TSTR) is used to observe whether the generated data is favorable for the classification task, i.e., supervised model training is performed using the generated data and model efficacy is calculated using the real data.

First, the converged MMDs under GAN and DCGAN are compared and the results are shown in Table 1. It can be seen that for the optimal parameter (σ) of MMD under CTTS-GAN and GAN training, CTTS-GAN has the lowest MMD in both, indicating that it is more like sampling from real data distribution than data generated by other methods. As for the optimal σ trained under GAN, CTTS-GAN also works well compared to GAN, thus CTTS-GAN can facilitate the generation of data with real characteristics for adversarial networks. In addition, under these σ, GARCH is optimal under the autoregressive model, while FFT-AS is better than AS, but the difference is not significant.

Table 1: MMD under different methods and σ

σ	Yule-Walker	Kalman	Garch	A-S	FFT-AS	FIN-GAN	DCGAN	CTTS-GAN
-3	0.71	0.71	0.73	0.69	0.63	0.56	0.52	0.51
-2	0.68	0.63	0.62	0.59	0.55	0.53	0.49	0.47
-1	0.54	0.6	0.5	0.47	0.42	0.49	0.45	0.43
1	0.51	0.49	0.45	0.44	0.42	0.47	0.43	0.4
2	0.69	0.66	0.65	0.55	0.49	0.49	0.43	0.41
3	1.38	1.16	1.12	0.77	0.55	0.5	0.45	0.42

Then, the statistics of the characteristics of the generated data under GAN and CTTS-GAN are given and the statistics are shown in Table 2. It can be seen that, in addition to the basic features, both GAN and CTTS-GAN

show the closest values to the real data in terms of the trend of the market, and thus the generated data are characterized by market features and highly similar.

Table 2: Statistics for generated data through GAN and CTTS-GAN

	GAN	CTTS-GAN
Linear unpredictable	Yes	Yes
Pachytonicity a	4.01	4.12
Cluster fluctuation	Yes	Yes
Leverage	Yes	Yes
$1 \leqslant K \leqslant 10$	Yes	Yes
Trend	10	11

Finally, in order to test whether the generated data can be used for subsequent analysis, we used Logistic, Support Vector Machine (SVM), Plain Bayes (NB) and Random Forest (RF) classifiers to perform the TSTR process, in which TRTR represents the real data training and testing process for benchmarking, the results are shown in Table 3. CTTS-GAN can “restore” the real data better than other methods, and the scores of TSTR on different classifiers are higher than other models. In the MMD-based parameter search training, compared with the conventional DCGAN method, CTTS-GAN can converge the MMD parameters more quickly and stably, which reflects its ability to fit the data distribution quickly. From the comparison of data distribution on the test set, the CTTS-GAN model also possesses the lowest maximum mean difference value, so the data distribution learned by the generator is more similar to the real data distribution.

Table 3: TSTR score for benchmark and CTTS-GAN generate data

	Classifier	GAN	CTTS-GAN	FINGAN	TSDIGAN
TSTR	Logistic	0.72	0.75	0.54	0.7
	SVM	0.8	0.8	0.76	0.76
	Naive Bayes	0.69	0.74	0.66	0.65
	RF	0.78	0.84	0.73	0.86
TRTR	Logistic	0.77	0.74	0.63	0.72
	SVM	0.84	0.83	0.72	0.8
	Naive Bayes	0.68	0.72	0.65	0.68
	RF	0.82	0.81	0.8	0.84

IV. C. Visualization of data distribution

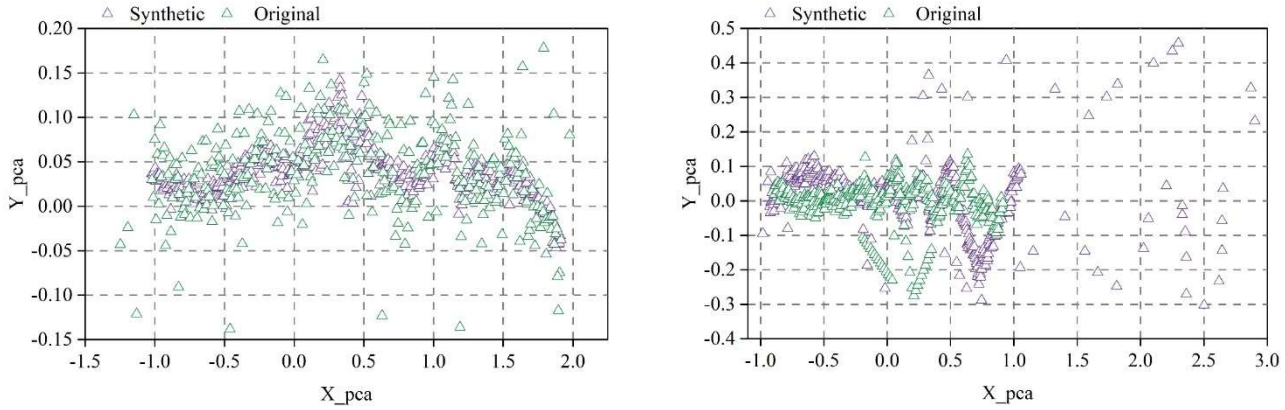
Data diversity is reflected in the fact that the financial time series data obtained through the generative model should contain the data characteristics of the real data, i.e., it should contain as many possible distributions of the real data as possible on the basis of their differences from the original real data distributions. In this section, PCA and T-SNE are used to visualize and analyze the data diversity by dimensionality reduction.

PCA is the principal component analysis technique. Principal Component Analysis, also known as Principal Component Analysis, aims to utilize the idea of dimensionality reduction to transform multiple metrics into a few composite metrics. In statistics, Principal Component Analysis PCA is a technique to simplify a data set. It is a linear transformation. The transformation transforms the data into a new coordinate system such that the first large variance of any data projection is on the first coordinate (called the first principal component), the second large variance is on the second coordinate (the second principal component), and so on. Principal component analysis is often used to reduce the dimensionality of a dataset while maintaining the features of the dataset that contribute the most to the variance, and is accomplished primarily by retaining the lower-order principal components and selectively ignoring the higher-order principal components. The low-order components thus obtained tend to preserve the most important component information of the original data.

t-SNE, t-distribution-stochastic neighborhood embedding, is a machine learning technique for data visualization and dimensionality reduction. The t-SNE works by considering the similarity between each data point and other data points. The similarities between high-dimensional data points are first computed, and then these similarities are mapped to a low-dimensional space while preserving the structure of the original similarities as much as possible. In this low-dimensional space, similar data points are mapped close to each other, while dissimilar data points are mapped relatively far away. Visualization and analysis of high-dimensional data can be achieved by t-SNE.

(1) Visualization and analysis of PCA distribution

The PCA distribution of the original GAN and CTTS-GAN generated data is shown in Fig. 4. (a) and (b) represent the comparison of PCA distribution of GAN model and CTTS-GAN model, respectively. The purple area in the figure indicates the distribution of model-generated data PCA after dimensionality reduction, and the green area indicates the distribution of real data PCA after dimensionality reduction. It can be clearly seen that the generated data of the original GAN covers a smaller area of the real data after PCA dimensionality reduction, and the overall discrete degree of the two is higher and the distribution area is wider. The above distribution indicates that the data generated by the GAN model cannot effectively respond to the data diversity of the real data. The generated data based on the CTTS-GAN model covers a larger area of the real sample data compared to the GAN generated data, and the distribution of CTTS-GAN generated data and real data is more dense, with a lower degree of overall dispersion; and the distribution area of Fig. b is more centralized compared to that of Fig. a, which reacts to the fact that the CTTS-GAN generated data has a certain degree of diversity of the real data.



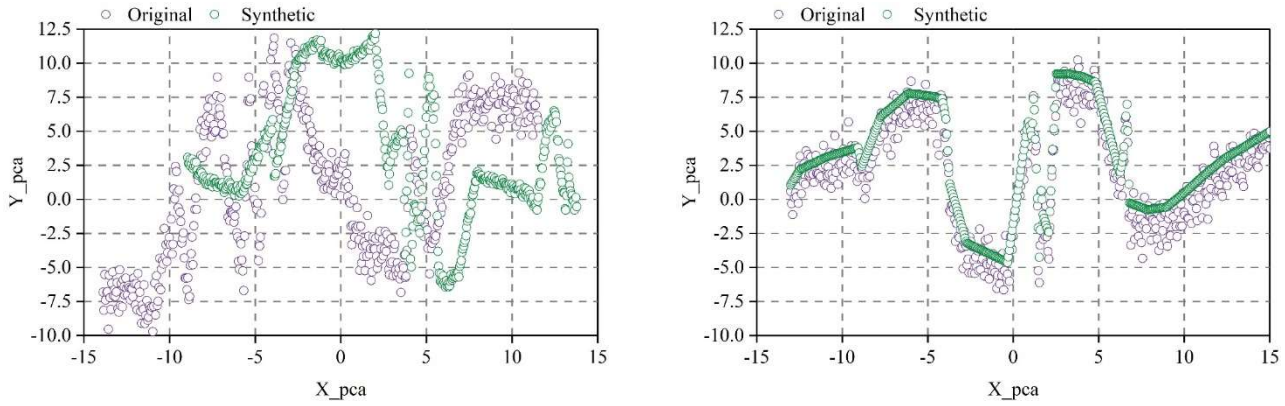
(a) GAN model PCA distribution

(b) CTTS-GAN model PCA distribution

Figure 4: Original GAN and CTTS-GAN generate data PCA distribution

(2) Visualization and analysis of t-SNE distribution

The distribution pairs of the original GAN and CTTS-GAN generated data and real sample data processed by the t-SNE method are shown in Fig. 5. (a) and (b) denote the comparison of t-SNE distributions of GAN and CTTS-GAN models, respectively. The green and purple area distributions indicate the distributions of the model-generated data and the real data after dimensionality reduction using t-SNE. By observing the t-SNE distribution of the original GAN and CTTS-GAN generated data and the real data, it can be found that the generated data obtained by using the original GAN has a more obvious interval portion in the two-dimensional spatial distribution with the real data, and the generated data fails to cover the distribution area of the real data well, and the degree of dispersion between the two is higher. At the same time, on the right side of Fig. a, there are more independent green areas, indicating that the generated data have lost the distribution characteristics of the real data in the region, and the generated data are not correlated with the real data, which shows that the generated data of the original GAN is poor in data diversity, and it fails to learn the distribution of the features embedded in the real data in the process of model training. Observing the t-SNE distribution of CTTS-GAN generated data in Fig. b, it can be clearly found that the generated data basically cover the possible distribution of the real sample data, and the distribution of the two distributions is more dense than that of Fig. a. And there is only a case where the individual generated distributions are detached from the distribution of the real sample data. In summary, CTTS-GAN generated data can better reflect the diversity of the real data, implying the data characteristics of the real sample data, which also indicates that the CTTS-GAN model performs better than the original GAN model in generating financial time series.



(a) GAN model t-SNE distribution

(b) CTTS-GAN model t-SNE distribution

Figure 5: The t-SNE distribution of raw GAN and CTTS-GAN generated data

In summary, under the same number of iterations, the model of CTTS-GAN has the best generation effect, and the generated data contains information closer to the real data, which can replace the real data to some extent for model prediction and classification research.

V. Conclusion

The CTTS-GAN model in this study exhibits superior performance over the traditional GAN model under multiple evaluation metrics. In the Maximum Mean Difference (MMD) evaluation, the MMD value of CTTS-GAN is 0.42, which is significantly lower than that of other methods, indicating that its generated data is closer to the distribution of real data. In the classification task, the data generated by CTTS-GAN achieves a TSTR score of 0.83 when trained using Support Vector Machine (SVM), which is higher than that of traditional GAN at 0.80. This indicates that CTTS-GAN not only effectively generates data with real characteristics, but also provides better application performance in the subsequent analysis. In addition, the advantages of CTTS-GAN in generating financial time series data are especially prominent, and it can better capture the market trend and volatility characteristics in time series data, which has strong practical application value.

Funding

Teaching Research Project of Anhui University of Chinese Medicine: Exploration and Research on the Construction Path of PBL Teaching Mode Based on AIGC (2024xjjy_yb028).

Anhui Provincial Teaching Research Project: Research on the Database Blended Teaching Model Oriented by Capability Cultivation in the Perspective of New Engineering (2022jyxm869).

References

[1] Wilson, S. J. (2017). Data representation for time series data mining: time domain approaches. Wiley Interdisciplinary Reviews: Computational Statistics, 9(1), e1392.

[2] Barandas, M., Folgado, D., Fernandes, L., Santos, S., Abreu, M., Bota, P., ... & Gamboa, H. (2020). TSFEL: Time series feature extraction library. SoftwareX, 11, 100456.

[3] Schneider, T., Helwig, N., & Schütze, A. (2017). Automatic feature extraction and selection for classification of cyclical time series data. tm-Technisches Messen, 84(3), 198-206.

[4] Fulcher, B. D., & Jones, N. S. (2017). hctsa: A computational framework for automated time-series phenotyping using massive feature extraction. Cell systems, 5(5), 527-531.

[5] Kim, S., Chung, E., & Kang, P. (2023). FEAT: A general framework for feature-aware multivariate time-series representation learning. Knowledge-Based Systems, 277, 110790.

[6] Ashraf, M., Anowar, F., Setu, J. H., Chowdhury, A. I., Ahmed, E., Islam, A., & Al-Mamun, A. (2023). A survey on dimensionality reduction techniques for time-series data. IEEE Access, 11, 42909-42923.

[7] Dong, Y., Qin, S. J., & Boyd, S. P. (2021). Extracting a low-dimensional predictable time series. Optimization and Engineering, 1-26.

[8] Wang, S., Lin, Y., Jia, Y., Sun, J., & Yang, Z. (2024). Unveiling the multi-dimensional spatio-temporal fusion transformer (MDSTFT): A revolutionary deep learning framework for enhanced multi-variate time series forecasting. IEEE Access.

[9] Baldán, F. J., & Benítez, J. M. (2023). Complexity measures and features for times series classification. Expert Systems with Applications, 213, 119227.

[10] Rhif, M., Ben Abbes, A., Farah, I. R., Martínez, B., & Sang, Y. (2019). Wavelet transform application for/in non-stationary time-series analysis: A review. Applied sciences, 9(7), 1345.

[11] Chen, Y., Ding, F., & Zhai, L. (2022). Multi-scale temporal features extraction based graph convolutional network with attention for multivariate time series prediction. Expert Systems with Applications, 200, 117011.
[12] Lindemann, B., Müller, T., Vietz, H., Jazdi, N., & Weyrich, M. (2021). A survey on long short-term memory networks for time series prediction. Procedia Cirp, 99, 650-655.
[13] He, Y., Seng, K. P., Ang, L. M., Peng, B., & Zhao, X. (2025). Hyper-CycleGAN: A new adversarial neural network architecture for cross-domain hyperspectral data generation. Applied Sciences, 15(8), 4188.
[14] Li, Z., Tao, R., Wang, J., Li, F., Niu, H., Yue, M., & Li, B. (2021). Interpreting the latent space of gans via measuring decoupling. IEEE transactions on artificial intelligence, 2(1), 58-70.
[15] Zhang, Z., Zhong, S. H., Fares, A., & Liu, Y. (2022). Detecting abnormality with separated foreground and background: Mutual generative adversarial networks for video abnormal event detection. Computer Vision and Image Understanding, 219, 103416.
[16] Cassisi Carmelo,Aliotta Marco,Cannata Andrea,Pistagna Fabrizio,Prestifilippo Michele,Torrisi Mario & Montalto Placido. (2024). TSDSystem: a framework to collect, archive and share time series data at volcanological observatories. Bulletin of Volcanology,86(8),
[17] Paulo Henrique Ranazzi,Xiaodong Luo & Marcio Augusto Sampaio. (2024). Improving the training performance of generative adversarial networks with limited data:Application to the generation of geological models. Computers and Geosciences,193,105747-105747.
[18] Wang Changgang,Cao Yu,Zhang Shi & Ling Tong. (2021). A Reconstruction Method for Missing Data in Power System Measurement Based on LSGAN. Frontiers in Energy Research,9,
[19] Nguyen Duc Thuan. (2024). Robust knowledge transfer for bearing diagnosis in neural network models using multilayer maximum mean discrepancy loss function. Measurement Science and Technology,35(12),126129-126129.