DCT-GAN: Dilated Convolutional Transformer-Based GAN for Time Series Anomaly Detection

Yifan Li, Xiaoyan Peng, Jia Zhang, Zhiyong Li, *Member, IEEE*, and Ming Wen

Abstract—Time series anomaly detection (TSAD) is an essential problem faced in several fields, e.g., fault detection, fraud detection, and intrusion detection, etc. Although TSAD is a crucial problem in anomaly detection, few solutions in anomaly detection are suitable for it at present. Recently, some researchers use GAN-based methods such as TAnoGAN and TadGAN to solve TSAD problem. However, problems such as model collapse, low generalization capability and poor accuracy still exist. In this article, we proposed a Dilated Convolutional Transformer-based GAN (DCT-GAN) to enhance accuracy and improve generalization capability of the model. Specifically, DCT-GAN utilize several generators and a single discriminator to alleviate the mode collapse problem. Each generator consists of a dilated convolutional neural network and a Transformer block to obtain fine-grained and coarse-grained information of the time series, which is a useful component to improve generalization capability. We also use weight-based mechanism to balance these generators. Experiments verify the effectiveness of our method and each part of DCT-GAN.

Index Terms—Anomaly detection, time series analysis, dilated convolutional neural network, transformer, generative adversarial network, weight-based mechanism

1 Introduction

Time series is a sequence of numerical data points in successive order. Time series anomalies refer to the observations which are especially different from others in a specific time sequence. Detecting anomalies in time series plays an important role in many situations, e.g., extreme weather or climate change [1], network intrusion [2], chemical process [3], electric network [4], etc. Undetected anomalies may cause economic losses and even human casualties. However, the label of time series is usually too difficult or expensive to obtain. As the result, using unsupervised learning technique to develop appropriate anomaly detection methods is imminent.

Researchers have already been devoted to detect anomalies in time series for a few decades. Earlier work tried to establish a mathematical model which is fit to the given data perfectly, and regard those samples that do not fit the model as anomalies. These methods distinguish normal and abnormal samples by measuring the distance between each sample [5] or density at each point [6]. In order to achieve good experimental results, we need to find a model that can perfectly fit the real data, but it is hard to describe data in the real world using a single model when the situation is complicated and the data is affected by multiple factors. As the result, traditional methods are outmoded and deep learning becomes prevalent in detecting anomalies.

As the deep learning techniques matures, researchers make great progress and attain better performance in solving anomaly detection problem than traditional ways. Schreyer *et al.* handled fraud detection problem using Autoencoder (AE)-based method [7] by calculating reconstruction error. Munir *et al.* proposed DeepANT [8] model which is mainly based on CNN for time series data. Cho *et al.* use Variant Autoencoder (VAE) [9] to solve TSAD problem. Wu *et al.* [10] developed a prediction-driven, unsupervised anomaly detection scheme, which adopts a backbone model combining the decomposition and the inference of time series data. Liu *et al.* [11] proposed an automatic Quasi-periodic time series (QTS) anomaly detection framework consisting of a two-level clustering-based QTS segmentation algorithm and a hybrid attentional LSTM-CNN model.

In recent years, generative adversarial network (GAN) has been proved to be one of the most effective methods to solve anomaly detection problem, and carried out better performance than the previous deep learning methods. A series of GAN-based solutions such as AnoGAN [12], EGBAD [13] and GANomaly [14] have been proposed. Unfortunately, few of them are suitable for sequential data. Although some new methods try to fill this gap, these methods still encounter problems like lack of accuracy and robustness.

In this article, we introduce a novel method called Dilated Convolutional Transformer GAN (DCT-GAN), that will be demonstrated to be effective and stable to deal with

- *Yifan Li, Xiaoyan Peng, Jia Zhang, and Zhiyong Li are with the College of Computer Science and Electronic Engineering, Hunan University, Changsha, Hunan 410082, China. E-mail: {yifanli, pxy18892336880, zhangjia1992, zhiyong.li}@hnu.edu.cn.*
- *Ming Wen is with the State Grid Hunan Electric Power Company Limited Economical & Technical Research Institute, Hunan Key Laboratory of Energy Internet Supply-demand and Operation, Changsha, Hunan 410004, China. E-mail: hnjyy1903_wen@126.com.*

Manuscript received 20 Mar. 2021; revised 29 Oct. 2021; accepted 21 Nov. 2021. Date of publication 24 Nov. 2021; date of current version 7 Mar. 2023.
This work was supported in part by the National Key Research and Development Program of China under Grant 2018YFB1308604, National Natural Science Foundation of China under Grant 61976086, State Grid Science and Technology Project under Grant 5100-202123009A, and Special Project of Foshan Science and Technology Innovation Team under Grant FS0AA-KJ919-4402-0069.
(Corresponding author: Zhiyong Li.)
Recommended for acceptance by V. Moreira.
Digital Object Identifier no. 10.1109/TKDE.2021.3130234

TSAD problem. Specifically, the main contribution of our work are as follows:

- We propose an architecture called DCT-GAN to enhance accuracy and robustness when dealing with TSAD problem.
- We design multiple Transformer-based generators with different scales to get both coarse-grained and fine-grained information in our GAN-based model for improving its generalization capability.
- We use weight-base mechanism to integrate the generators in order to make the model compatible with different kinds of anomalies.

The remainder of this article is organized as follow. The previous works aimed at solving TSAD problem are reviewed in Section 2. Section 3 will introduce our methodology DCT-GAN to solve TSAD problem. Section 4 mainly confirms the effectiveness of our work by experiments. Finally in Section 5, we conclude the whole article.

2 Related Work

In this section, we briefly review previous related work on anomaly detection problem, especially GAN-based algorithm, which are closely related to our work. We also introduce some other advanced time series anomaly detection for comparison.

2.1 GAN-Based Anomaly Detection Algorithm

GAN was initially invented for image generation tasks. Dating back to 2014, vanilla GAN [15] was first proposed by Goodfellow *et al.* initially for sample generation. In 2016, DCGAN [16] was proposed by Radford *et al.* to improve the performance and stability of vanilla GAN especially for image generation. Both the discriminator and generator of DCGAN use CNN to replace the multi-layer perceptron in GAN. At the same time, in order to make the entire network differentiable, the pooling layer in CNN is removed, and the fully connected layer is replaced by a global pooling layer to reduce the amount of computation. Later, researchers began to apply GAN on time series. In 2017, Yoon *et al.* proposed TimeGAN [17] to create a generative model for generating time series data, this method combines themes from auto-regressive models for sequence prediction, GAN-based methods for sequence generation, and time series representation learning. In the same year, Esteban *et al.* proposed RCGAN [18] to produce realistic real-valued multidimensional time series (especially medical data), it make use of recurrent neural networks (RNNs) in the generator and the discriminator.

GAN-based method was first used for anomaly detection in 2017 , Schlegl *et al.* successfully applied DCGAN in AnoGAN [12] to identify the anomalous optical coherence tomography (OCT) images, pioneering the application of GAN in anomaly detection field. However, AnoGAN is time-consuming because the parameter tuning occurs every time when a new photo is received. Therefore, in 2018, Zenati *et al.* proposed EGBAD [13] to solve this problem. They added an encoder to map the real data to the latent space for joint training, so that the model can omit the time for parameter tuning. Based on EGBAD, Akcay *et al.* proposed GANomaly [14] in 2019. GANomaly re-encode the generated samples and compare the result to the latent representation produced by the previous encoder, and it is proved to have a higher confidence level than EGBAD. Later in the same year, they further proposed a Skip-GANomaly model [19] by adding skip-connection to GANomaly for alleviating the vanishing gradient problem. In 2020, Sohn *et al.* proposed a two-stage framework [20] for one-class classification, the author trained a deep neural network to obtain a high-level data representation in the first stage, and built a one-class classifier using representations from the first stage in the second stage. These methods have achieved good results in the image field at that time, however, they are not suitable for pure time series anomaly detection as the method and tricks are designed for images.

Recently in 2020, researchers began to explore the application of GAN in time series anomaly detection. Bashar *et al.* proposed TAnoGAN [21] which combines LSTM and AnoGAN to handle the problem, and it is the first attempt of GAN to detect anomalies in time series data as far as we know. Later, in September 2020, TadGAN [22] was proposed by Geiger *et al.* They leverage GANs Generator and Critic to compute robust anomaly scores at every time step and get a better performance than baseline methods in anomaly detection.

In a word, although GAN is used to detect anomalies for several years, it is still an emerging field for detecting anomalies in time series.

2.2 Other Advanced Time Series Anomaly Detection Algorithm

Besides GAN-based methods, plenty of other solutions are proposed by researchers who devoted to detect anomalies in time series. They study in different perspective of this problem and have already made great contributions.

Some researchers try to enhance the accuracy of detecting anomalies. For example, in 2019, Munir *et al.* proposed DeepANT [8] to find point anomalies, contextual anomalies, and discords in time series data using CNN-based network and exceed some traditional algorithms. In 2020, Cheng *et al.* proposed HS-TCN [23] model to detect anomalies in IoT communication data, it is a semi-supervised model so it can train unlabeled data based on a small number of labeled data to weed out uncertain records during IoT communication.

However, sometimes there is not enough data to establish the model, or the data is varying in a short period of time so that the pre-trained model is not satisfied with the current situation. Therefore, there are also some researchers mainly focusing on the real-time performance of the model. In 2021, Maciag *et al.* proposed OeSNN-UAD [24], it has been improved on the basis of the evolving Spiking Neural Networks, and OeSNN-UAD can further learn in the process of classifying input values to be anomalous or not, so the model can be dynamically adjusted in real time.

3 Preliminaries

3.1 Vanilla GAN

Vanilla GAN is mainly composed of a generator network G and a discriminator network D. Both of the networks play a minimax two-player game until reach the Nash Equilibrium. Generator G tries to learn the distribution of the real data, and discriminator D turns to distinguish the real data from the generated fake data. The value function of vanilla GAN can be calculated as the following equation:

$$\begin{aligned}&\min_G \max_D V(D,G)\\&= \mathbb{E}_{\boldsymbol{x}\sim p_{data}(\boldsymbol{x})}[\log D(\boldsymbol{x})] + \mathbb{E}_{z\sim p_z(z)}[\log(1-D(G(z)))],\end{aligned}$$

and the model can be trained according to this equation. Stochastic gradient descent optimizer is used to update G and D. While updating D, ascending the stochastic gradient

$$\nabla_{\theta_d}\frac{1}{m}\sum_{i=1}^{m}[\log D(\boldsymbol{x}^{(i)}) + \log(1-D(G(\boldsymbol{z}^{(i)})))],$$

where m refers to the number of instances sampled from the original data in each iteration while training. Then while updating G, descending the stochastic gradient

$$\nabla_{\theta_g}\frac{1}{m}\sum_{i=1}^{m}\log(1-D(G(\boldsymbol{z}^{(i)}))).$$

According to Goodfellow's view, the ideal state of training vanilla GAN is making G recover the training data distribution and making D no longer discriminative.

3.2 Dilated CNN

Dilated CNN [25] is initially designed for dense prediction problems such as semantic segmentation. It modifies vanilla CNN by adding zeros into convolution kernels, so that CNN is able to expand its reception field. According to the initial paper, dilated CNN can systematically aggregate multi-scale contextual information without losing resolution. In this article, we use the model to capture different scale of features.

3.3 Transformer

Transformer [26] is a powerful model in Nature Language Processing (NLP) which is initially designed for translation task, and soon it was widely used in various tasks in NLP. Recently, Transformer has also achieved outstanding performance in image processing. One of the main contribution of Transformer is incorporating self-attention mechanism into the neural network. Therefore, in this article, we use the encoder block of Transformer to extract features in time series.

4 Methodology

This section formally introduces our DCT-GAN algorithm. First we give a specific statement of TSAD problem and define the problem formally. Then we introduce the overall framework of our DCT-GAN and the basic ideas behind the framework. Finally, we go into more details about our method. The description of notations can be found in Appendix, which can be found on the Computer Society Digital Library at http://doi.ieeecomputersociety.org/10.1109/TKDE.2021.3130234.

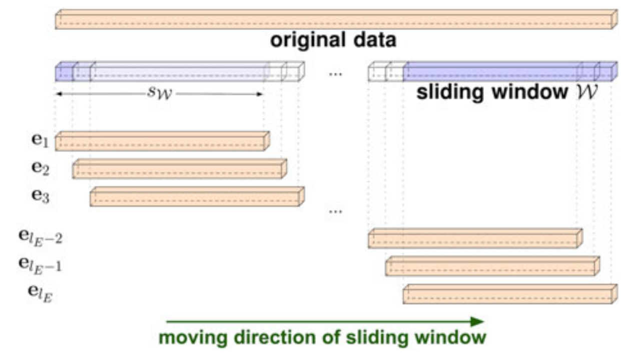

Fig. 1. The relationship between original data and sliding window. With the sliding window continue to slide in the positive direction of the time axis, we can intercept constantly changing data from the original data, and combine these data fragments into a new dataset for anomaly analysis.

4.1 Problem Statement

Time series anomaly detection (TSAD) problem is to find out time segments that have different patterns from the others in the observed time series. In Section 4, we propose the DCT-GAN method to solve this problem.

Due to the continuity of time series, we use sliding windows to split the original data into data fragments, and combine these fragments into a new dataset for anomaly analysis. Formally, we define this problem as follows.

Given a time series $X := (x_1, x_2, \ldots, x_{l_X}) \in \mathbb{R}$, we extract a set of small sequences $E := \{\mathbf{e}_1, \mathbf{e}_2, \ldots, \mathbf{e}_{l_E}\}$ from X using a sliding window $\mathcal{W}$ with window size $s_{\mathcal{W}}$ and time step size 1. Obviously, $l_E = l_X - s_{\mathcal{W}} + 1$. Fig. 1 shows some details about how to use sliding windows to split the original data. Our final goal is to find out a set of anomaly sequence $A := \{\mathbf{a}_1, \mathbf{a}_2, \ldots, \mathbf{a}_{l_A}\}, A \subset E$ which contains l_A anomaly sequences.

4.2 Overall Framework

The overall framework of DCT-GAN is depicted in Fig. 2, the whole framework can be divided into two stages: the model training stage and the anomaly detecting stage. The basic idea of DCT-GAN is to improve the robustness, generalization capability and accuracy when handling TSAD problem. Dilated CNN is used to capture multiple scales of features on a certain detection window to enhance the robustness, Transformer-based Network is utilized to organize a series of detection windows, and multi-scale generators are used to generate fake data from different perspectives to ensure the generalization capability of the model. We will discuss in more detail in the following part.

4.2.1 Framework of Model Training Stage

Fig. 2a shows the model training stage. The whole architecture of model training stage is GAN-based with multiple generators and a single discriminator. Each generator learns information from a different size of detection window by using Dilated CNN to extract features, then a Transformer-based Neural Network is followed to handle with the extracted information. After integration, discriminator receives the integrated fake information from the generators and the corresponding real sequences, and then learns to distinguish them.

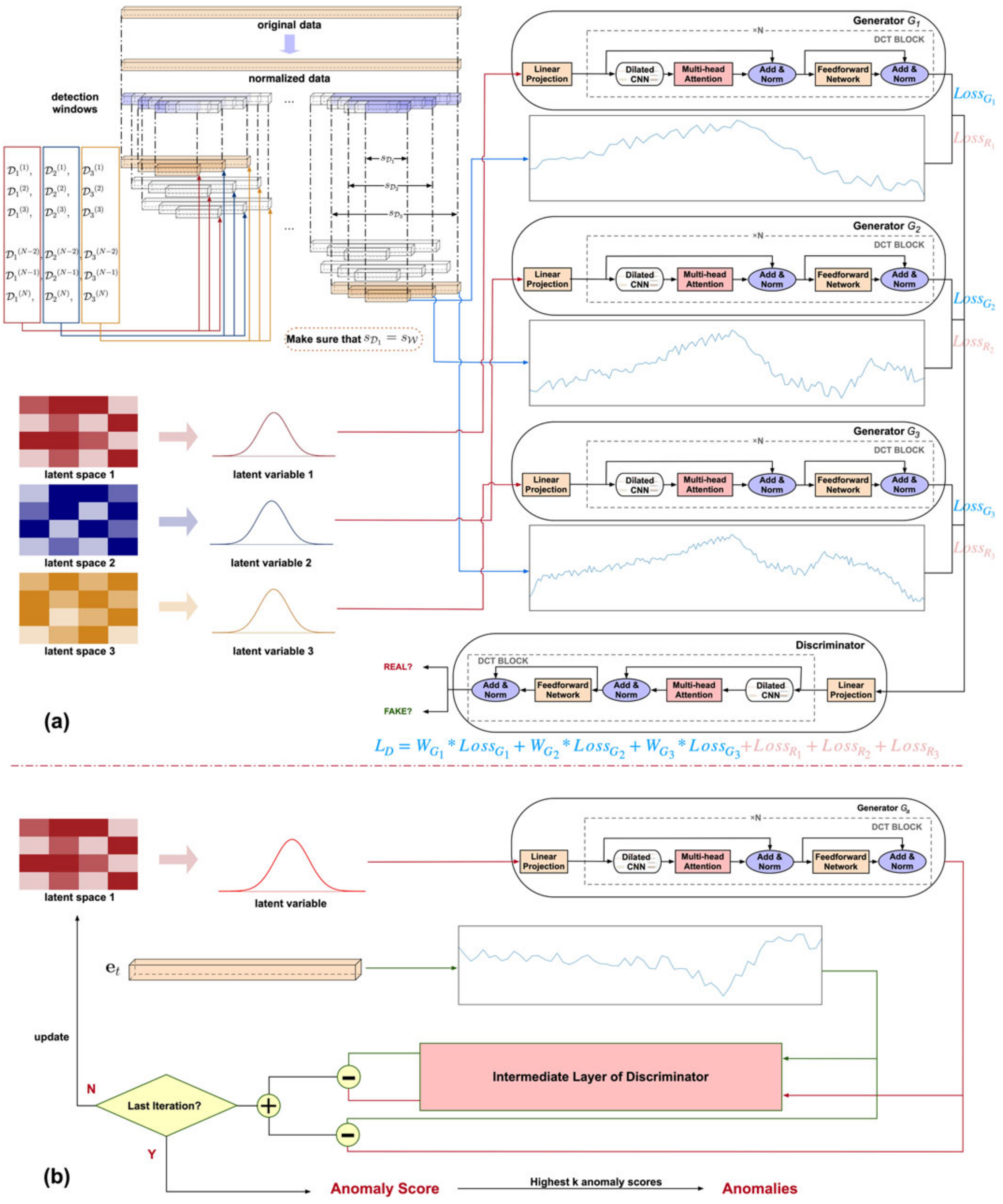

Fig. 2. The overall framework of DCT-GAN. (a) shows the model training stage of our method, the generators and discriminator is trained in this stage. Upper-left corner shows the procedure of data preprocessing, lower-left corner shows the latent space and latent variables, generators and discriminator are on the right side of the figure. The loss function shows our weighted-method in the Model Training Stage.(b) reveals the anomaly detecting stage of our method, we use the well-trained generator G_a to train the latent variable and further calculate the anomaly score. (The structure of Dilated CNN block is depicted in Fig. 3.)

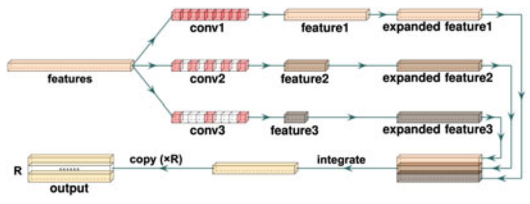

Fig. 3. The structure of Dilated CNN block.

We further explain the framework of training stage of DCT-GAN in details. First, we need to normalize the original data for preprocessing. The original time series X is a sequence including l_X data, and we define a set of detection windows $\mathbb{D}$. For a certain generator G_i, the raw data is cut by the detection window $\mathcal{D}_i \in \mathbb{D}$ into small sequences, and then delivered to the next step. q detection windows in $\mathbb{D}$ are used to cut out q time slices of different lengths $(s_{\mathcal{D}_1}, s_{\mathcal{D}_2}, \ldots, s_{\mathcal{D}_q})$ from the normalized data. At time t, the obtained q time slices are $\mathcal{D}_t^{(1)}, \mathcal{D}_t^{(2)}, \ldots, \mathcal{D}_t^{(q)}$. Notice that the first detection window should have the same size of the sliding window $\mathcal{W}$.

Then, we establish q generators to generate fake time slices of length $s_{\mathcal{D}_1}, s_{\mathcal{D}_2}, \ldots, s_{\mathcal{D}_q}$. The set of generators are denoted by $\mathbb{G} = \{G_1, G_2, G_3, \ldots G_q\}$ respectively. For each generator G_i, the corresponding Dilated Convolutional Neural Network (Dilated CNN) and Transformer-based Network is defined as C_i and T_i separately, and the size of related detection window, i.e., the maximum length of sequence that G_i can receive at the same time, is obviously denoted by $\mathcal{D}_i$.

Finally, the q generators are integrated with different weight in conformity with their significance to generate the fake data, and a single discriminator D is used to discriminate the integrated fake data against the real data.

The generators and the discriminator have the same architecture. They are all composed of a linear projection module and a DCT (Dilated Convolutional Transformer) block. Linear projection is a widely-used method, it is mainly used for adjusting the dimension of data to ensure the input dimension matches our model. Transformer is a powerful tool to handle with text sequences, we use Transformer because it is adept at handling sequential data. but it is hard for Transformer to get information from the latent space. Therefore, we propose a Dilated Convolutional Transformer (DCT) architecture to solve this problem. The DCT block is a Transformer-based architecture that contains a Dilated CNN module, and the structure of Dilated CNN module is depicted in Fig. 3. The input feature is fed into three dilated convolutional neural networks of different dilation, then we expand these features to make sure they have the same length. After integration, the feature is copied R times and finally output. The core portion of the Transformer-based network is self-attention technique. Self-attention maps a query $\mathbf{Q}$ and a set of key-value pairs $\mathbf{K}$ and $\mathbf{V}$ to an output. For the generator G_i, $\mathbf{Q}_i, \mathbf{K}_i, \mathbf{V}_i \in \mathbb{R}^{w \times d}$ where d is the dimension of latent representation in Transformer. and the following equation holds at the time given timestamp t:

$$\mathbf{Q}_i{}^{(t)} = \mathbf{K}_i{}^{(t)} = \mathbf{V}_i{}^{(t)} = \mathrm{D}CB_i(l(\mathcal{D}_i{}^{(t)})), \tag{1}$$

where $l(\cdot)$ is a set of linear projections, $\mathcal{D}_i{}^{(t)}$ is the sequence detected by the detection window $\mathcal{D}_i$ at t moment, and $\mathrm{D}CB_i$ refers to the dilated CNN block in the generator G_i. Then the self-attention block can be formed as

$$\mathrm{Att}(\mathbf{Q}_i{}^{(t)}, \mathbf{K}_i{}^{(t)}, \mathbf{V}_i{}^{(t)}) = (\mathbf{D}_i{}^{(t)})^{-1}\mathbf{A}_i{}^{(t)}\mathbf{V}_i{}^{(t)},$$

where

$$\begin{aligned} \mathbf{A}_i{}^{(t)} &= \exp\left(\frac{\mathbf{Q}_i{}^{(t)}\mathbf{K}_i{}^{(t)\top}}{\sqrt{d}}\right), \\ \mathbf{D}_i{}^{(t)} &= \mathrm{diag}(\mathbf{A}_i{}^{(t)}\mathbf{1}_{s_{\mathcal{D}_i}}). \end{aligned} \tag{2}$$

$\mathbf{1}_{s_{\mathcal{D}_i}}$ is all 1 vector of length $s_{\mathcal{D}_i}$.

For a certain Dilated CNN block $\mathrm{D}CB_i$ in generator G_i, let k_i^j refers to the jth convolution kernel in G_i, $\hat{k_i^j}$ refers to convolution kernel after dilation. The relationship among convolution kernel size $|k_i^j|$, dilated convolution kernel size $|\hat{k_i^j}|$, stride s_i^j, dilation rate r_i^j and padding p_i are satisfied with the formulas $|\hat{k_i^j}| = 2 * p_i + 1$ and $r_i^j = s_i^j = (|\hat{k_i^j}| - 1)/(|k_i^j| - 1)$. Formally, let $\hat{x}$ refers to sequence x with padding, the mth element of the result of $\mathrm{D}ilatedConv_{i,j}(x)$ can be formulated as

$$\begin{aligned} \mathrm{DilatedConv}_{i,j}(x)[m] &= \left(\sum_{n=\hat{m}}^{|\hat{k_i^j}|+\hat{m}} \hat{x}[n] * \hat{k}[n]\right) \\ \text{where} \quad \hat{m} &= \left\lfloor \frac{m-1}{r_i^j} \right\rfloor + 1. \end{aligned} \tag{3}$$

We use multiple generators in our model mainly to alleviate the impact of mode collapse. Mode collapse is a classical GAN failure occurs while training GAN. It means that each iteration of generator over-optimizes for a particular discriminator, and the discriminator never manages to learn its way out of the trap [27]. Therefore, when mode collapse happens, the generator can just generate certain kinds of samples among real data, and the generalization capability of model is limited. From the perspective of Optimal Transport Theory, the root cause of mode collapse is that deep neural network can only approximate continuous mapping, while the transmission mapping is a discontinuous mapping. In other words, the target mapping is not in the representable functional space of deep neural network. Therefore, a single generator is limited to generate all types of samples. To alleviate mode collapse, the model should be able to approximate discontinuous mapping. Apparently, using multiple generators is an effective way to solve the problem. To enforce different generators to produce different types of samples, we feed these generators with different lengths of sequences, so that the generators can learn from different perspectives. This illustrate why DCT-GAN is less vulnerable to mode collapse. From another perspective, if there is only one generator, the window size will have a huge impact on the results and reduce the robustness of the model. Specifically, if the size of detection window $s_{\mathcal{D}_i}$ is set up to a small number, the mode may become powerless because of the limited information received while meeting long-term context anomaly, but if $s_{\mathcal{D}_i}$ is set up to a number that is too large, the model may get too much useless information, then the model might suffer from poor accuracy or efficiency.

Multiple generators are useful for solving mode collapse problem. However, how to make full use of each generator is still a crucial problem. DCT block is capable to learn from a certain detection window with its multi-scale feature extraction and time series handling ability. We adopt multi-scale Dilated CNN for multi-scale Feature Extraction. A Dilated CNN has been used in semantic segmentation to expand receptive field [25] without loss of resolution. In our work, we retain the advantage of Dilated CNN and use it to get multi-scale information of the detection window. As we mainly focus on 1-dimension time series, we choose 1-dimension dilated CNN to process the data. Besides, to ensure the generators get the same amount of information at the same time, we make the multiple dilated CNN networks between each generator maintain a stable receptive field, this method encourages the generator to get equivalent amount of information at the same time. Moreover, in order to match the following Transformer-based network, we also pad the detection windows to make the size of their outputs equal.

4.2.2 Framework of Anomaly Detecting Stage

Fig. 2b shows the anomaly detecting stage. This stage is mainly used for giving out anomaly scores and finally output the detected anomalies.

Due to the small amount of abnormal samples, the well-trained model in the model training stage should be able to generate fake samples closed to the real normal data. Therefore, we use one of the generators to reconstruct the real normal data. Since the selected generator should be able to produce time series to simulate the real normal data, we choose G_a for anomaly detection (ensure that $s_{\mathcal{D}_a} = s_{\mathcal{W}}$, in this article we select G_1 as G_a). As G_a is a fixed structure from the well-trained DCT-GAN, we can back-propagate the latent variable z to find a proper latent variable z^* in the latent space that $G_a(z)$ is more similar to the real normal samples. Finally, We use G_a as generator, D as discriminator to establish a new GAN architecture for anomaly detection.

4.3 Method

In this part, we introduce the method of DCT-GAN. Like the framework part, we will introduce the method in model training stage and anomaly detecting stage respectively.

4.3.1 Method in Model Training Stage

First, we introduce the integration method for the generators. To balance these generators while calculating the discriminator loss, we need to weight each generator. As the generators have their own bias towards different kinds of anomalies, it is not a proper way to give each generator a fixed weight. Consequently, we use a weight-based method to dynamically adjust their weights while calculating the loss of the discriminator in this stage. Simply, we use the loss value while training in the previous iteration as the evidence of the importance of a certain generator. In details, suppose the loss of discriminator, i.e., BCE loss, when using generator G_u is larger than using G_v in a certain iteration $\mathcal{I}-1$, and $G_u, G_v \in \mathbb{G}$, then we consider the generator G_u is more satisfied with the real data as it gives a more accurate judgement of the data in this iteration, and we increase the weight of G_v in the next iteration $\mathcal{I}$. We define the initial weight ${W_{G_i}}^{(0)} = 1, \forall i \in \{1, 2, \ldots, q\}$. We calculate the weight of G_i in iteration $\mathcal{I}$ by the following equations:

$$
{W_{G_i}}^{(\mathcal{I})} = \frac{1/{L_{G_i,f}}^{(\mathcal{I}-1)}}{\sum_{n=1}^{q}(1/{L_{G_n,f}}^{(\mathcal{I}-1)})}, \tag{4}
$$

where $L_{G_i,f}^{(\mathcal{I})}$ refers to the binary cross entropy loss between $D(G_i)$ and fake labels in $\mathcal{I}$th iteration.

Second, we introduce the updating method of the generators and the discriminator. Like vanilla GAN, while training DCT-GAN, we update discriminator D and generators $\mathbb{G}$ alternatively. In the $\mathcal{I}$th iteration, while training D, the discriminator is updated based on the equation

$$
\nabla_{\theta_d} \frac{1}{Mq} \sum_{i=1}^{q} \sum_{\mathcal{I}=1}^{M} [\log D(\boldsymbol{x}_i^{(\mathcal{I})}) + W_{G_i} \log(1 - D(G_i(z_i^{(\mathcal{I})})))]. \tag{5}
$$

While training G_i, we update generator G_i according to the following equation:

$$
\nabla_{\theta_{g_i}} \frac{1}{M} \sum_{\mathcal{I}=1}^{M} \log(1 - D(G_i(z_i^{(\mathcal{I})}))), \tag{6}
$$

where M refers to the maximum iteration of training DCT-GAN.

Algorithm 1. Model Training Stage of DCT-GAN

Input: Time series, X; A set of detection window, $\mathbb{D}$; Batch size when training DCT-GAN, b; Maximum number of iteration, M; Number of generators, q.
Output: A set of well-trained generators $\mathbb{G}$; A well-trained discriminator D.
Initialize generators $\mathbb{G}$, discriminator D, initial weight of each generator ${W_{G_i}}^{(0)}$

```
1:  for $\mathcal{I} = 1, \mathcal{I} < M, \mathcal{I}{+}{+}$ do
2:  for $i = 1, i < q, i{+}{+}$ do
3:        Sample $b$ sequences $\{\mathcal{S}_i^{(1)}, \mathcal{S}_i^{(2)}, \ldots, \mathcal{S}_i^{(b)}\}$ from $\mathcal{S}_i$
4:          Calculate training loss (BCE loss) ${L_{G_i,f}}^{(\mathcal{I}-1)}$ for fake
            data of the previous iteration on each generator $G_i$
5:        Calculate ${W_{G_i}}^{(\mathcal{I})}$ according to Eq. (4)
6:      end for
7:      Update discriminators $D$ according to Eq. (5)
8:      for $i = 1, i < q, i{+}{+}$ do
9:        Update generators $G_i$ according to Eq. (6)
10:     end for
11: end for
```

Third, we introduce the loss function of DCT-GAN. The loss function for DCT-GAN mainly include two parts: GAN loss and gradient penalty. GAN loss describes how accurate the discriminator is to distinguish the generated data and real data, and gradient penalty [28] is used for enforce the Lipschitz constraint and make the model easier to converge. The complete loss function of model training stage in $\mathcal{I}$th iteration for DCT-GAN can be defined as follows:

TABLE 1
Categories in NAB Dataset [30]

Category	Description
realTweets	A collection of Twitter mentions of large publicly-traded companies such as Google and IBM. The metric value represents the number of mentions for a given ticker symbol every 5 minutes.
realTraffic	Real time traffic data from the Twin Cities Metro area in Minnesota, collected by the Minnesota Department of Transportation. Included metrics include occupancy, speed, and travel time from specific sensors.
realKnownCause	This is data for which we know the anomaly causes, including real cases e.g., taxi schedules or CPU utilization
realAWSCloudwatch	AWS server metrics as collected by the AmazonCloudwatch service. Example metrics include CPU Utilization, Network Bytes In, and Disk Read Bytes.
realAdExchange	Online advertisement clicking rates, where the metrics are cost-per-click (CPC) and cost per thousand impressions (CPM). One of the files is normal, without anomalies.
artificialWithAnomaly	Artificial time series with anomalies
artificialNoAnomaly	Artificial data without anomalies

$$\begin{aligned}\mathcal{L}_{\mathrm{DG}} &= \frac{1}{Mq}\sum_{i=1}^{q}\sum_{\mathcal{I}=1}^{M}\log D(\boldsymbol{x}_i{}^{(\mathcal{I})}) \\ &+ \frac{1}{Mq}\sum_{i=1}^{q}\sum_{\mathcal{I}=1}^{M} W_{G_i}{}^{(\mathcal{I})}\log\left(1 - D(G_i(z_i^{(\mathcal{I})}))\right) \\ &+ \lambda\sum_{i=1}^{q}\left(\left\|\nabla_{\tilde{\boldsymbol{x}}_i} D(\tilde{\boldsymbol{x}}_i^{(\mathcal{I})})\right\|_2 - 1\right)^2,\end{aligned} \tag{7}$$

where $\tilde{\boldsymbol{x}}_i^{(\mathcal{I})} = \mu\boldsymbol{x}_i^{(\mathcal{I})} + (1-\mu)G_i(z_i^{(\mathcal{I})})$, $\boldsymbol{x}_i^{(\mathcal{I})}$ refers to the real data used in the $\mathcal{I}$th iteration.

Algorithm 2. Anomaly Detecting Stage of DCT-GAN

Input: Iteration, T; Weight allocation parameter in loss function, ϕ; Sequence for anomaly detection, E; The well-trained generator which is related to a detection window whose window size is $s_{\mathcal{W}}$, G_a; The prior probability of generating noise compatible with G_a; Anomaly rate η.
Output: The detected list of anomalies in E, A.

```
1: for $i = 1, i <= T, i{+}{+}$ do
2:     Train the latent space $z$ using $E$ to minimize Eq. (8)
3:     if $i == T$ then
4:         Store the well-trained latent space $z_a = z$
5:     end if
6: end for
7: Calculate the value of $\mathcal{L}_{TSAD}$ when $z = z_a$
8: Sort sequences $E$ by loss $\mathcal{L}_{\mathrm{TSAD}}$, and select the top $l_E \cdot \eta$ samples to form the anomaly list $A$
```

Finally, The pseudo code of *Model Training Stage of DCT-GAN* is given in Algorithm 1.

4.3.2 Method in Anomaly Detecting Stage

We briefly introduce the loss function in anomaly detecting stage. The loss function for anomaly detection is composed of two parts: the loss between real data and fake data, and the loss between feature extracted from the discriminator that is provided with real data and fake data. Complete loss function of anomaly detecting stage can be expressed by the following equation:

$$\begin{aligned}\mathcal{L}_{\mathrm{TSAD}} &= (1-\phi)\cdot\sum|\boldsymbol{x} - G_a(z^*)| \\ &+ \phi\cdot\sum|f(\boldsymbol{x}) - f(G_a(z^*))|,\end{aligned} \tag{8}$$

where $\phi \in (0,1)$ and $f(\cdot)$ refers to the output of an intermediate layer of the discriminator. Obviously, our loss function is composed of two parts: the reconstruction loss and the discriminator-based loss.

The pseudo code of *Anomaly Detecting Stage of DCT-GAN* is given in Algorithm 2.

5 Experiments

In this section, we introduce the dataset, show the result of ablation experiment, and compare the performance of our DCT-GAN with baseline method and state-of-the-art methods.

5.1 Experiment Setup

5.1.1 Datasets

We evaluate the performance of our method through Numenta Anomaly Benchmark (NAB) [29], which can be obtained in [30]. NAB dataset is composed of 58 sub-datasets, covering 7 categories [24] including *realTweets*, *realTraffic*, *realKnownCause*, *realAWSCloudwatch*, *realAdExchange*, *artificialWithAnomaly*, *artificialNoAnomaly*, (more details can be found in Table 1). The sub-datasets in each category are as follows:

- *realTweets: Twitter Volume AAPL, Twitter Volume AMZN, Twitter Volume UPS, Twitter Volume CRM, Twitter Volume CVS, Twitter Volume FB, Twitter Volume GOOG, Twitter Volume IBM, Twitter Volume KO, Twitter Volume PFE.*
- *realTraffic: Occupancy 6005, Occupancy T4013, Speed 6005, Speed 7578, Speed T4013, TravelTime 387, TravelTime 451.*
- *realKnownCause: Ambient Temperature System Failure, CPU Utilization ASG Misconfiguration, EC2 Request Latency System Failure, Machine Temperature System Failure, NYC Taxi, Rogue Agent Key Hold, Rogue Agent Key Updown.*
- *realAWSCloudwatch: EC2 CPU Utilization 5f5533, EC2 CPU Utilization 24ae8d, EC2 CPU Utilization 53ea38, EC2 CPU Utilization 77c1ca, EC2 CPU Utilization 825cc2, EC2 CPU Utilization ac20cd, EC2 CPU Utilization c6585a, EC2 CPU Utilization fe7f93, EC2 Disk write bytes 1ef3de, EC2 Disk write bytes c0d644, EC2 Network in 5abac7, EC2 Network in 257a54, ELB Request count*

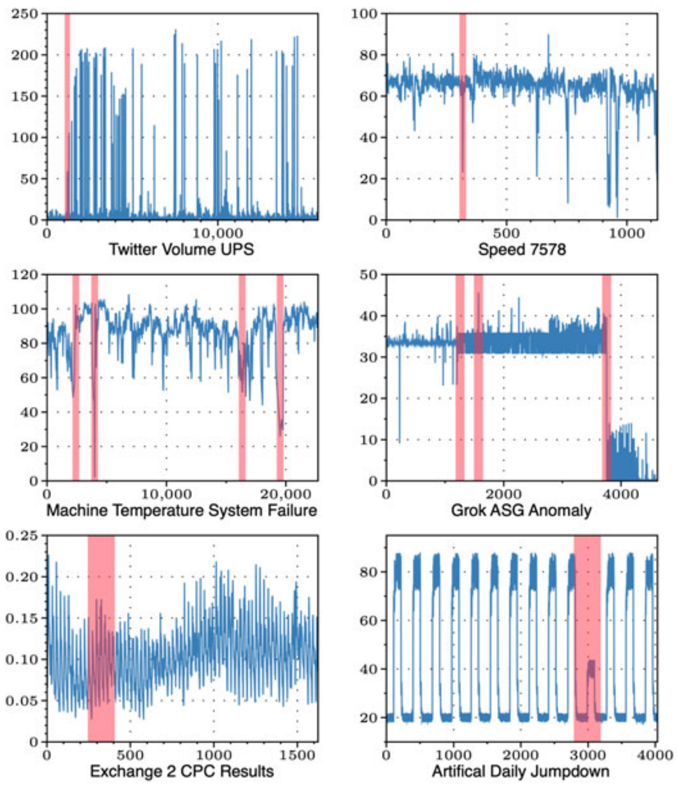

Fig. 4. Some samples in the NAB dataset. We select one sub-dataset from each category except *artificialNoAnomaly* as there is no anomaly in it. blue lines refer to the data flow and red ranges refer to the anomaly windows.

8c0756, Grok ASG Anomaly, RDS CPU Utilization cc0c53, RDS CPU Utilization e47b3b.

- *realAdExchange: Exchange 2 CPC Results, Exchange 2 CPM Results, Exchange 3 CPC Results, Exchange 3 CPM Results, Exchange 4 CPC Results, Exchange 4 CPM Results.*
- *artificialWithAnomaly: Art Daily Flatmiddle, Art Daily Jumpsdown, Art Daily Jumpsup, Art Daily Nojump, Art Increase Spike Density, Art Load Balancer Spikes.*

We apply our method on the previous six categories (as *artificialNoAnomaly* does not contain anomaly samples). The whole dataset contains a total of 365,551 data points, and each sub-dataset is a 1-dimension time series containing 1,127 to 22,695 data points, we visualize the data and anomalies of some sub-datasets from the NAB dataset in Fig. 4. Due to the diversity of the NAB dataset, it is possible for us to verify the generalization capability of our model.

We also apply our method on multi-dimensional time series because they are very common in real scene. SWAT [31] and WADI [32] datasets are used in this experiment.

SWAT dataset is proposed to support research in the design of secure Cyber Physical Systems. It is a scaled down version of a real-world industrial water treatment plant producing 5 gallons per minute of water filtered via membrane based ultrafiltration and reverse osmosis units. This plant allowed data collection under two behavioral modes: normal and attacked. We regard the attacked periods as anomalies in our experiment. There are 449,919 data points in SWAT dataset.

WADI consists of three stages controlled by PLCs and two stages controlled via RTUs. Each PLC and RTU uses sensors to estimate the system state and the actuators to effect control. We also regard the attacked periods as anomalies in our experiment. WADI dataset contains 172801 data points.

5.1.2 Evaluation Metrics

We use AUROC and F1-Score to measure the performance of these models. First, we use the anomaly score and real labels to evaluate AUROC. Then, according to the anomaly score and anomaly percentage η, we can determine which samples should be regarded as anomalies, and we compare the detected anomalies with real labels to evaluate TP, FP, TN, FN samples and further evaluate F1-Score.

In this article, we measure the performance based on two metrics: F1-Score and AUROC (Area Under Receiver Operating Characteristic curve). First, we use the anomaly score and real labels to evaluate AUROC, AUROC computes area under the ROC curve from anomaly score. The value of $\mathcal{L}_{TSAD}$ when $z = z_a$ in Algorithm 2 is regarded as the anomaly score. Then, according to the anomaly score and anomaly percentage η, we can determine which samples should be regarded as anomalies, and we compare the detected anomalies with real labels to evaluate TP, FP, TN, FN samples and further evaluate F1-Score. F1- Score is calculated by the following formula:

$$F_1 = \frac{2 * precision * recall}{precision + recall}, where \\ precision = \frac{TP}{TP + FP}, \quad recall = \frac{TP}{TP + FN}. \tag{9}$$

TP (True Positive), FP (False Positive) and FN (False Negative) are obtained by comparing the anomaly list A and the corresponding real anomaly labels.

5.1.3 Experiment Setup and Parameters

The experiments in this article mainly consist of two parts: ablation experiment and comparison experiment. We use a subset of the NAB dataset (including 12 sub-datasets ,that means, extract 2 sub-datasets for each category) for ablation experiment, and full dataset (52 sub-datasets) for the comparison experiment.

To balance the time complexity and detection accuracy, we empirically use 3 generators in DCT-GAN, and set the window size of their related detection windows as $s_{\mathcal{D}_1} = 60$, $s_{\mathcal{D}_2} = 120$, and $s_{\mathcal{D}_3} = 240$. The size of the sliding window is set to $s_{\mathcal{W}} = 60$, which is equal to $s_{\mathcal{D}_1}$, so we can use the first generator as G_a in Algorithm 2. The value of anomaly rate η is set to a number in $\{5\%, 10\%, 15\%, 20\%\}$, so we can obtain the performance of our method when different extent of outliers are defined as anomalies. More information about setting parameters is shown in Table 2.

All experiments are performed on a computer equipped with RTX 2080Ti GPU. The source code of DCT-GAN is written in pytorch.

5.2 Ablation Experiment

We conduct ablation study to show the necessity of each component in our model. In this experiment we present

TABLE 2
Parameters in Our Experiments

Parameter	Value
$s_{\mathcal{W}}$	$s_{\mathcal{W}} = 60$
$s_{\mathcal{D}_i}$	$s_{\mathcal{D}_1} = 60, s_{\mathcal{D}_2} = 120, s_{\mathcal{D}_3} = 240$
q	$q = 3$
w	$w = 3$
s_i^j	$s_i^1 = 1, s_i^2 = 2, s_i^3 = 4 (i \in 1, 2, 3)$
r_i^j	$r_i^1 = 1, r_i^2 = 2, r_i^3 = 4 (i \in 1, 2, 3)$
p_i	$p_i = 4$
b	$b = 32$
M	$M = 20$
d	$d = 8192$
T	$T = 50$
η	η is a number in $\{5\%, 10\%, 15\%, 20\%\}$

different subsets of our method for anomaly detection. The settings are specified as follows.

- *GAN + Transformer:* We simply use Transformer as generator and discriminator of GAN to detect anomalies. This model is capable of detecting anomalies based on the reconstructive loss and dealing with time series using the Transformer model.
- *GAN + DCT block:* We use a DCT block the single generator and single discriminator of GAN. The vanilla CNN is a dilated CNN with dilation rate of 1, therefore, using DCT block can extract more macroscopic information from a given time series.
- *Our Method:* The full model introduced before is used for anomaly detection. Our model use multiple generators to avoid mode collapse, we also make further efforts to balance each generator using weight-based mechanism and add gradient penalty to the loss function.

The result of ablation experiment demonstrates the effectiveness of our model. As is depicted in Fig. 5, using GAN with the DCT block lead to a better result than using GAN with vanilla Transformer. That means it is too rigid to simply piece together the GAN and Transformer modules, which will result in low detection accuracy, Using DCT block is helpful to increase the accuracy. This is because DCT block add a dilated CNN neural network in the generator, which is helpful to map the latent space to a new dimensions, and make it easy for the generator to generate fake samples. Our method further improves the performance of the model, as we can generate fake samples from different perspective in each generator while using multiple generators, and then filter out the most effective generator through the weight-based mechanism.

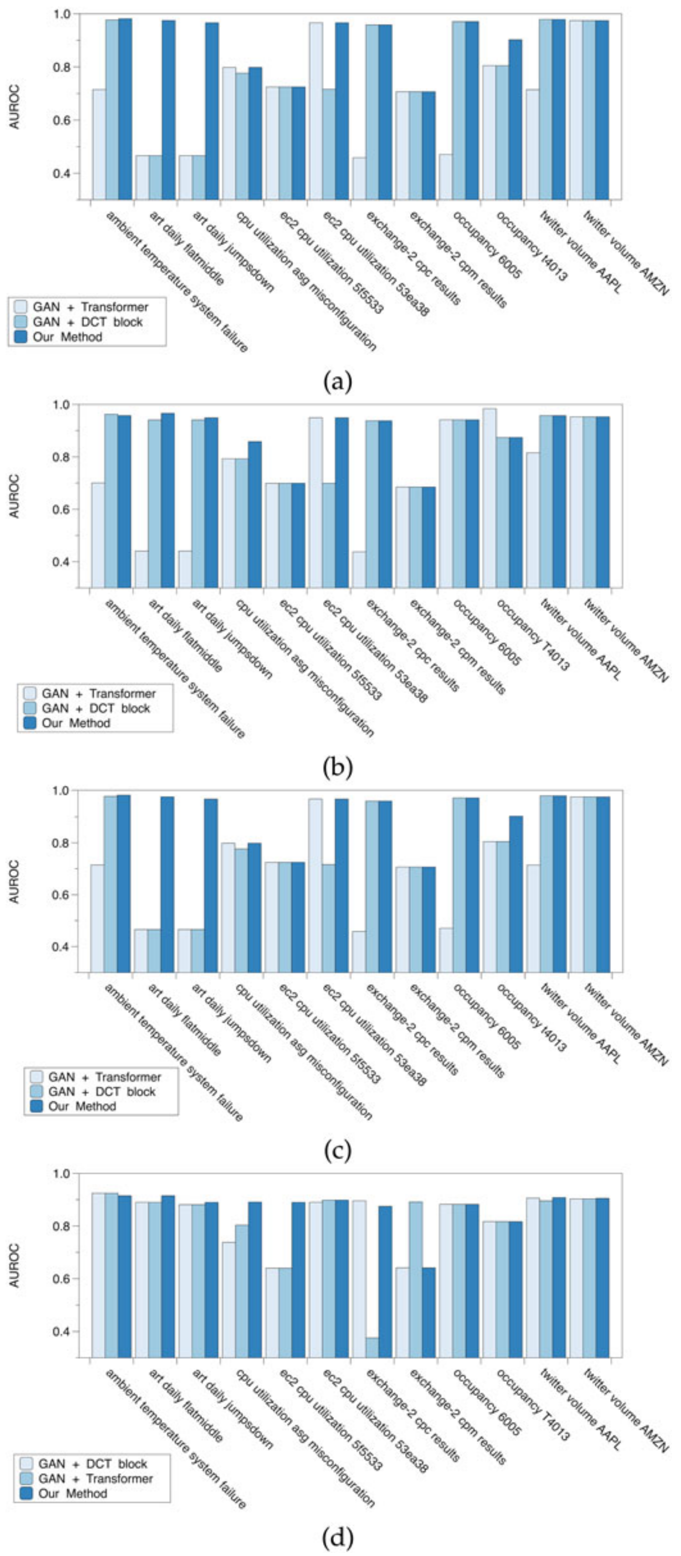

Fig. 5. Result of ablation experiment when $\eta = 5\%, 10\%, 15\%$ and 20%. The performance of GAN with DCT block is better than that of GAN with Transformer, and our final model DCT-GAN is even better. This result also proof that our method has better ability to avoid mode collapse.

5.3 Comparison Experiment

We compare our final model with some baselines and state-of-the-art methods.

5.3.1 *Baselines*

- *LSTM [33]:* LSTM is a useful and fundamental neural network to handle with sequential data, it is a special recurrent neural network using a forget gate, an input gate and an output gate to achieve long term memory. Two LSTM layers with 80 units each and a subsequent dense layer are used in the experiment.
- *DeepANT [8]:* DeepANT use CNN block as a predictor to predict the data in the next timestamp according to the previous several timestamps, and compare

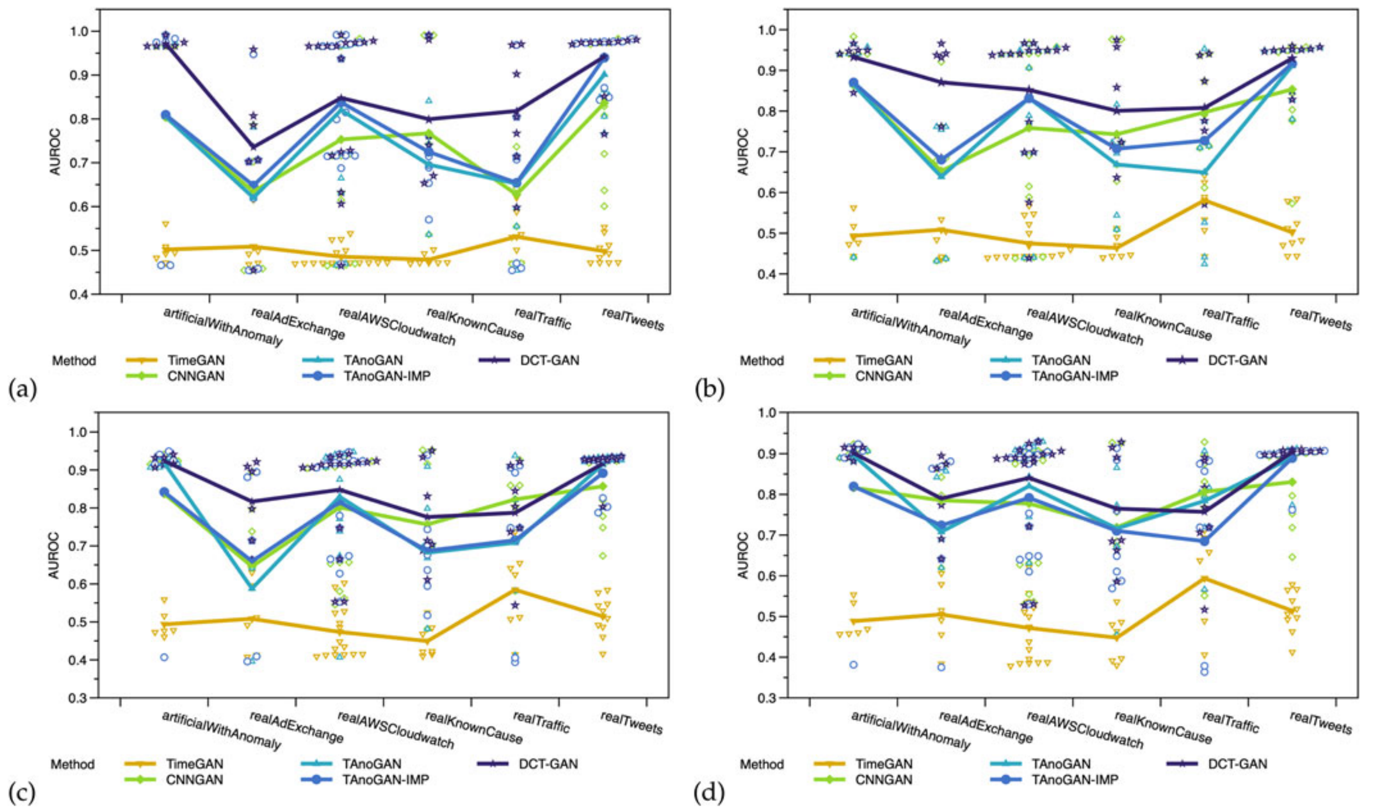

Fig. 6. The result of comparison experiment when $\eta = 5\%$, $\eta = 10\%$, $\eta = 15\%$ and $\eta = 20\%$. Each hollow point refers to a result of a sub-dataset in comparison experiment, and each solid point refers to the average AUROC of all sub-datasets in a certain category. Our DCT-GAN has achieved great performance in most cases.

the predicted data with real data to measure if abnormal events happen.

- *OeSNN-UAD [24]:* OeSNN-UAD is based on evolving Spiking Neural Networks (eSNN), but unlike typical eSNN architecture, neurons in the output repository of our architecture are not divided into known a prior decision classes. OeSNN-UAD is already realized on the NAB dataset and the average F1-Score has been made public, so we directly compare it with our results.

5.3.2 State-of-the-Art Methods

- *CNNGAN:* We use a GAN-based network whose generator and discriminator are consist of three CNN blocks, and LeakyRELU is used as activation function, which is similiar to the model in AnoGAN [12].
- *TAnoGAN [21]:* TAnoGAN is a method designed for anomaly detection on one-dimension time series, and originally tested on the NAB dataset according to their paper. The code of TAnoGAN is open-sourced, so we just recalculate AUROC on the whole dataset using their code.
- *TAnoGAN-IMP:* In this experiment, we improve the TAnoGAN model and add a 4 fully connected layers to the generator and discriminator respectively, the activation function of discriminator is set to be ELU, and LeakyRELU in generator. The experimental result shows that it performs better than original TAnoGAN.
- *TimeGAN [17]:* TimeGAN is a popular method initially designed for generating multi-dimensional time series, it combines themes from auto-regressive models for sequence prediction, GAN-based methods for sequence generation, and time series representation learning. In this experiment, we calculated the euclidean distance between the generated data and normal data to determine whether the sample is abnormal.
- *TadGAN [22]:* TadGAN uses a generator, a discriminator and an encoder in its model. Gradient penalty is also added to its loss function. TadGAN is originally tested on the NAB dataset as well, but it is difficult to reproduce on their code, so we use their result of F1-Score to compare with ours directly.
- *MAD-GAN [35]:* MAD-GAN is a GAN-based multivariate time series anomaly detection model. MAD-GAN uses the Long-Short-Term-Memory Recurrent Neural Networks (LSTM-RNN) as the base models in the GAN framework to capture the temporal correlation of time series distributions.
- *GDN [36]:* GDN is an attention-based graph neural network approach, it is mainly composed of four parts: Sensor Embedding, Graph Structure Learning, Graph Attention-Based Forecasting, and Graph Deviation Scoring. GDN tries to observe the connection between data obtained from different sensors.

In this experiment, we use all sub-datasets in NAB that contains anomalies (totally 52 datasets). When $\eta = 5\%$, The result of AUROC is shown in Fig. 6a, and F1-Score is shown in Table 3, weighted average is calculated by the following

TABLE 3
F1-Score Result of Comparison Experiment (Methods With "*" Indicates That the Result is From the Corresponding Literature, Which is Derived From [22], [24], and [34], "-" Indicates That Original Paper Does Not Provide the Result on This Dataset. We Set $\eta = 5\%$ When Testing on Other Method)

	LSTM*	DeepANT*	OeSNN-UAD*	CNNGAN	TAnoGAN	TAnoGAN-IMP	TadGAN*	**DCT-GAN**
realTweets	0.543	0.075	0.310	0.654	0.751	**0.799**	0.609	**0.799**
realTraffic	0.634	0.223	0.340	0.492	0.509	0.635	0.486	**0.659**
realKnownCause	-	0.200	0.324	0.580	0.460	0.524	-	**0.646**
realAWSCloudwatch	0.474	0.146	0.369	0.676	**0.711**	0.696	0.644	0.709
realAdExchange	0.538	0.132	0.234	0.690	0.528	0.576	**0.800**	0.638
artificialWithAnomaly	0.375	0.156	0.427	0.773	0.684	0.725	**0.800**	0.741
Weighted Average	0.509	0.149	0.339	0.647	0.635	0.674	0.653	**0.707**

formula:

$$WeightedAverage = \frac{\sum_i^6 |Category_i| * F1Score_i}{\sum_i^6 |Category_i|}, \tag{10}$$

where $Category$ refers to the set of categories mentioned in Table 1 and $|Category| = \{10, 7, 7, 17, 6, 6\}$.

In Fig. 6, we compare our method with some GAN-based algorithms. Each point in the figure stands for a result of a sub-dataset. We can clearly discover that although the red points (representing DCT-GAN) and other points are distributed roughly in the same area when AUROC $>= 0.6$, we can rarely find red points when AUROC < 0.6, this is an evidence that our method has better generalization capability. The solid lines reveals the average performance of each method, and DCT-GAN has almost the best performance in all 6 fields.

In Table 3, we further compare our DCT-GAN with more methods proposed recently. We can find that DCT-GAN has the highest weighted average F1-Score. TadGAN has better performance on realAdExchange and artificialWithAnomaly, This is probably because TadGAN concentrate on mode collapse too much so that the model has weak performance on the datasets that are not easy to suffer from mode collapse.

From the chart and table we present, we can draw the conclusion that our method perform better than other method proposed recently.

Since the strictness of abnormality determination is divergent in different scenarios, we also need to study the influence of the proportion of anomalies on our proposed algorithm. As the result, we design an experiment to observe whether the performance of the model changes with the value of η. In Figs. 6b, 6c, and 6d, the performance of each method when $\eta = 10\%$, $\eta = 15\%$ and $\eta = 20\%$. We discover that although the value of AUROC changes with η, the accuracy ranking of each method hardly changes, and our DCT-GAN always performs well. Moreover, their AUROC grows while η increases because more anomalies is detected, so we can choose an appropriate η for a certain task when detecting anomalies, e.g., we can choose high η when dealing with industrial fault diagnosis tasks because one single fault might cause catastrophic consequences, but we should appropriately reduce the value of η while

TABLE 4
Training Time, Parameters as well as GPU Memory Usage of DCT-GAN

	Dataset	Samples in Training Set	Training Time Per Sample(second)	Parameters	GPU Memory Usage(MiB)
	Exchange 4 CPC Results	1642	0.1197	52034	6907
	Twitter Volume AAPL	15901	0.1387	52034	6293
	Art Daily Flatmiddle	4031	0.1344	52034	5951
DCT-GAN	Ambient Temperature System Failure	7207	0.1336	52034	6547
	Occupancy 6005	2320	0.1306	52034	7591
	EC2 CPU Utilization 5f5533	3972	0.1321	52034	5953
	Average	5846	0.1315	52034	6540
	Exchange 4 CPC Results	1642	0.0816	143790	10369
	Twitter Volume AAPL	15901	0.0091	143790	10375
	Art Daily Flatmiddle	4031	0.0335	143790	10369
TimeGAN	Ambient Temperature System Failure	7207	0.0190	143790	10375
	Occupancy 6005	2320	0.0589	143790	10377
	EC2 CPU Utilization 5f5533	3972	0.0345	143790	10369
	Average	5846	0.0394	143790	10372
	Exchange 4 CPC Results	1642	0.0274	170326	1077
	Twitter Volume AAPL	15901	0.0262	170326	1079
	Art Daily Flatmiddle	4031	0.0266	170326	1077
TAnoGAN	Ambient Temperature System Failure	7207	0.0265	170326	1077
	Occupancy 6005	2320	0.0275	170326	1079
	EC2 CPU Utilization 5f5533	3972	0.0274	170326	1077
	Average	5846	0.0269	170326	1078

TABLE 5
F1-Score of Methods in Multi-Dimensional Time Series Experiments (We Set $\eta = 5\%$ in This Experiment)

Dataset	Method	F1-Score
SWAT	**DCT-GAN**	0.78
	MAD-GAN	0.77
	GDN	0.81
WADI	**DCT-GAN**	0.35
	MAD-GAN	0.37
	GDN	0.57

handling data denoising tasks, because if the η is too large, normal samples would be filtered out and the dataset would be distorted.

Though experiments have already proofed the effectiveness of DCT-GAN, our method still encountered the time-consuming problem. In Table 4, we reported the training time, the size of the system both in number of parameters as well as GPU memory usage of DCT-GAN, and compared with TimeGAN and TAnoGAN. The result shows that the model training stage of our method is quite time consuming, however, the anomaly detecting stage does not take long, we can pre-train the model offline and detect anomalies online to overcome this problem.

5.3.3 Experiment on Multi-Dimensional Data

Experiments have proved the effectiveness of the DCT-GAN method on one-dimensional data. However, in many cases, time series data is multi-dimensional. Therefore, it would be important to test whether DCT-GAN can be extended and generalized to multi-dimensional time series dataset.

We use Principal Component Analysis (PCA) method to reduce the dimension of the original data, convert the multi-dimensional data into single-dimensional data, and then use DCT-GAN to detect anomalies. Table 5 shows the performance of DCT-GAN on the SWAT and WADI datasets. We can find that the performance of DCT-GAN in multi-dimensional data is acceptable (close to MAD-GAN).

6 Conclusion

In conclusion, we proposed a novel method called DCT-GAN, to solve the time series anomaly detection problem successfully. We utilize Transformer to process the time series, GAN-based model to reconstruct the samples and find out anomalies, dilated CNN structure to extract information from the latent space, multiple generators to deal with the mode collapse problem. The experiments demonstrate the necessity of each part of our DCT-GAN and the effectiveness of the model.

The experiments on the NAB dataset show that DCT-GAN is more accurate than most other methods, and more stable than all these methods. In future work, we will further slim the network to solve the time-consuming problem.

References

[1] B. Barz, E. Rodner, Y. G. Garcia, and J. Denzler, "Detecting regions of maximal divergence for spatio-temporal anomaly detection," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 41, no. 5, pp. 1088–1101, May 2019.

[2] L. Heng and T. Weise, "Intrusion detection system using convolutional neuronal networks: A cognitive computing approach for anomaly detection based on deep learning," in *Proc. IEEE 18th Int. Conf. Cogn. Informat. Cogn. Comput.*, 2019, pp. 34–40.

[3] X. Yang and D. Feng, "Generative adversarial network based anomaly detection on the benchmark tennessee eastman process," in *Proc. 5th Int. Conf. Control Autom. Robot.*, 2019, pp. 644–648.

[4] A. Anwar and A. N. Mahmood, "Anomaly detection in electric network database of smart grid: Graph matching approach," *Electr. Power Syst. Res.*, vol. 133, pp. 51–62, 2016.

[5] D. J. Weller-Fahy , B. J. Borghetti, and A. A. Sodemann, "A survey of distance and similarity measures used within network intrusion anomaly detection," *IEEE Commun. Surveys Tuts.*, vol. 17, no. 1, pp. 70–91, First Quarter 2015.

[6] M. M. Breuniq, H. P. Kriegel, R. T. Ng, and J. Sander, "LOF: Identifying density-based local outliers," *ACM SIGMOD Rec.*, vol. 29, no. 2, pp. 93–104, 2000.

[7] M. Schreyer, T. Sattarov, D. Borth, A. Dengel, and B. Reimer, "Detection of anomalies in large-scale accounting data using deep autoencoder networks," 2017, *arXiv:1709.05254*.

[8] M. Munir, S. A. Siddiqui, A. Dengel, and S. Ahmed, "DeepAnT: A deep learning approach for unsupervised anomaly detection in time series," *IEEE Access*, vol. 7, pp. 1991–2005, 2019.

[9] J. An and S. Cho, "Variational autoencoder based anomaly detection using reconstruction probability," *Special Lecture IE*, vol. 2, no. 1, pp. 1–18, 2015.

[10] W. Wu, L. He, W. Lin, Y. Su, and S. A. Jarvis, "Developing an unsupervised real-time anomaly detection scheme for time series with multi-seasonality," *IEEE Trans. Knowl. Data Eng.*, early access, Nov. 3, 2020, doi: 10.1109/TKDE.2020.3035685.

[11] F. Liu, X. Zhou, J. Cao, Z. Wang, and Y. Zhang, "Anomaly detection in quasi-periodic time series based on automatic data segmentation and attentional LSTM-CNN," *IEEE Trans. Knowl. Data Eng.*, early access, Aug. 6, 2020, doi: 10.1109/TKDE.2020.3014806.

[12] T. Schlegl, P. Seeböck, S. M. Waldstein, U. Schmidt-Erfurth , and G. Langs, "Unsupervised anomaly detection with generative adversarial networks to guide marker discovery," in *Proc. Int. Conf. Inf. Process. Med. Imag.*, 2017, pp. 146–147.

[13] H. Zenati, C. S. Foo, B. Lecouat, G. Manek, and V. R. Chandrasekhar, "Efficient GAN-based anomaly detection," 2018, *arXiv: 1802.06222*.

[14] S. Akcay, A. Atapour-Abarghouei, and T. P. Breckon, "GANomaly: Semi-supervised anomaly detection via adversarial training," in *Proc. Asian Conf. Comput. Vis.*, 2019, pp. 622–637.

[15] I. J. Goodfellow *et al.*, "Generative adversarial nets," in *Proc. 27th Int. Conf. Neural Inf. Process. Syst.*, 2014, pp. 2672–2680.

[16] A. Radford, L. Metz, and S. Chintala, "Unsupervised representation learning with deep convolutional generative adversarial networks," in *Proc. 4th Int. Conf. Learn. Representations*, 2016.

[17] J. Yoon *et al.*, "Time-series generative adversarial networks," in *Proc. 33rd Int. Conf. Neural Inf. Process. Syst.*, 2019, pp. 5508–5518.

[18] C. Esteban, S. L. Hyland, and G. Rätsch, "Real-valued (medical) time series generation with recurrent conditional gans," 2017, *arXiv: 1706.02633*.

[19] S. Akcay, A. Atapour-Abarghouei , and T. P. Breckon, "Skip-GANomaly: Skip connected and adversarially trained encoder-decoder anomaly detection," in *Proc. Int. Joint Conf. Neural Netw.*, 2019, pp. 1–8.

[20] K. Sohn, C. L. Li, J. Yoon, M. Jin, and T. Pfister, "Learning and evaluating representations for deep one-class classification," pp. 1–24, 2020.

[21] M. A. Bashar and R. Nayak, "TAnoGAN: Time series anomaly detection with generative adversarial networks," in *Proc. IEEE Symp. Series Comput. Intell.*, 2020, pp. 1778–1785.

[22] A. Geiger, D. Liu, S. Alnegheimish, A. Cuesta-Infante, and K. Veeramachaneni, "TadGAN: Time series anomaly detection using generative adversarial networks," in *Proc. IEEE Int. Conf. Big Data*, 2020, pp. 33–43.

[23] Y. Cheng, Y. Xu, H. Zhong, and Y. Liu, "Leveraging semisupervised hierarchical stacking temporal convolutional network for anomaly detection in IoT communication," *IEEE Internet of Things J.*, vol. 8, no. 1, pp. 144–155, Jan. 2021.

[24] P. S. Maciag, M. Kryszkiewicz, R. Bembenik, J. L. Lobo, and J. Del Ser , "Unsupervised anomaly detection in stream data with online evolving spiking neural networks," *Neural Netw.*, vol. 139, pp. 118–139, 2021.

[25] F. Yu and V. Koltun, “Multi-scale context aggregation by dilated convolutions,” in *Proc. 4th Int. Conf. Learn. Representations*, 2016, pp. 1–13.
[26] A. Vaswani *et al.*, “Attention is all you need,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5999–6009.
[27] *Google*, “Generative adversarial GAN,” Feb. 10, 2020. Accessed: Mar. 9, 2021. [Online]. Available: https://developers.google.com/machine-learning/gan/problems
[28] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, and A. Courville, “Improved training of wasserstein GANs,” in *Proc. 31st Int. Conf. Neural Inf. Process. Syst.*, 2017, pp. 5768–5778.
[29] S. Ahmad, A. Lavin, S. Purdy, and Z. Agha, “Unsupervised real-time anomaly detection for streaming data,” *Neurocomputing*, vol. 262, pp. 134–147, 2017.
[30] *Numenta*, “The numenta anomaly benchmark,” May 6, 2014. Accessed: Jan. 30, 2021. [Online]. Available: https://github.com/numenta/NAB
[31] J. Goh, S. Adepu, K. N. Junejo, and A. Mathur, “A dataset to support research in the design of secure water treatment systems,” in *Proc. Int. Conf. Critical Inf. Infrastructures Secur.*, 2017, pp. 88–99.
[32] C. M. Ahmed, V. R. Palleti, and A. P. Mathur, “WADI: A water distribution testbed for research in the design of secure cyber physical systems,” in *Proc. 3rd Int. Workshop Cyber-Physical Syst. Smart Water Netw.*, 2017, pp. 25–28.
[33] K. Hundman, V. Constantinou, C. Laporte, I. Colwell, and T. Soderstrom, “Detecting spacecraft anomalies using LSTMs and nonparametric dynamic thresholding,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Mining*, 2018, pp. 387–395.
[34] A. Lavin and S. Ahmad, “Evaluating real-time anomaly detection algorithms - The numenta anomaly benchmark,” in *Proc. IEEE 14th Int. Conf. Mach. Learn. Appl.*, 2015, pp. 38–44.
[35] D. Li, D. Chen, B. Jin, L. Shi, J. Goh, and S. K. Ng, “MAD-GAN: Multivariate anomaly detection for time series data with generative adversarial networks,” in *Proc. Int. Conf. Artif. Neural Netw.*, 2019, pp. 703–716.
[36] A. Deng and B. Hooi, “Graph neural network-based anomaly detection in multivariate time series,” *Proc. AAAI Conf. Artif. Intell.*, 2021, vol. 35, pp. 4027–4035.

Yifan Li received the BS and MS degrees from the Zhejiang University of Technology, China, in 2016 and 2019, respectively. He is working towards the PhD degree in the College of Computer Science and Electronic Engineering, Hunan University, China. His current research interests include deep learning and anomaly detection.

Xiaoyan Peng received the BS degree from Guizhou University, China, in 2020. She is currently working towards the MS degree in the College of Computer Science and Electronic Engineering, Hunan University, China. Her current research interests include deep learning and anomaly detection.

Jia Zhang received the MSc degree in mathematics and econometrics from Hunan University, Changsha, China, in 2014, where she is currently working toward the PhD degree in the College of Computer Science and Electronic Engineering. Her research interests include optimization algorithm, anomaly detection, and machine learning.

Zhiyong Li (Member, IEEE) received the MSc degree in system engineering from the National University of Defense Technology, Changsha, China, in 1996, and the PhD degree in control theory and control engineering from Hunan University, Changsha, China, in 2004. In 2004, he joined the College of Computer Science and Electronic Engineering, Hunan University, China, where he is currently a full professor. He has published more than 100 papers in international journals and conferences. His research interests include intelligent perception and autonomous moving body, machine learning and industrial big data, and intelligent optimization algorithms with applications. He is a member of China Computer Federation (CCF) and Chinese Association for Artificial Intelligence (CAAI).

Ming Wen received the MS and PhD degrees from Hunan University, China, in 2006 and 2010, respectively. He has been a senior engineer at State Grid Hunan Electric Power Company Limited Economical and Technical Research Institute. His research interests include energy internet demand forecasting, electricity price, smart grid, renewable energy accommodation and power system planning.