Neural Networks 191 (2025) 107770

Contents lists available at ScienceDirect

Neural Networks

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

Full Length Article

ITS2Graph: Graph-based generative adversarial learning for imbalanced time series classification

Chang Liu, Donghai Guan *, Weiwei Yuan, Çetin Kaya Koç

College of Computer Science and Technology, Nanjing University of Aeronautics and Astronautics, Nanjing, 210016, China

ARTICLE INFO

Keywords:
Time series classification
Class imbalance
Graph construction
High-order association
Generative adversarial learning

ABSTRACT

Time Series Classification (TSC) is a fundamental task in data mining and often suffers from class imbalance, particularly in real-world applications. Traditional methods often fail to capture high-order intrinsic dependencies among time series, especially when minority class samples are scarce. Effectively mining such associations to improve minority-class representation remains a significant challenge. To address this issue, we propose ITS2Graph, a graph-based generative adversarial learning framework that exploits high-order associations for imbalanced time series classification. An auto-encoder is employed to extract latent representations of time series, based on which pairwise similarities are computed to construct a graph, thereby reformulating TSC as a node classification task. To mitigate class imbalance, a graph generator synthesizes minority-class node features and their topological connections, while a Graph Convolutional Network (GCN) discriminator is trained to distinguish real from generated nodes. Experimental results on 22 real-world time series datasets demonstrate that ITS2Graph outperforms existing algorithms in imbalanced time series classification tasks.

1. Introduction

In the domains of data mining and machine learning, class imbalance has drawn considerable attention as a difficult problem in time series classification (Krawczyk, 2016b). The International Conference on Machine Learning (ICML) and the Association for the Advancement of Artificial Intelligence (AAAI) have both hosted seminars on this topic (He & Garcia, 2009), and a number of experts have even claimed that one of the top ten difficulties in data mining work is the imbalance issue (Yang & Wu, 2006). When there are significantly fewer instances in some classes than in others, the distribution of samples from various classes within a dataset is considered to be imbalanced (Gu, Angelov, & Soares, 2020). This situation is common in many real-world industrial applications, such as disease detection in medical diagnosis (Woods et al., 1993), fault prediction in industrial production (Mahadevan & Shah, 2009), and anomaly detection in the financial market (Waleed, Andrew, & John, 2022). Due to this imbalance, models may become biased in favor of the majority class during training, ignoring the characteristics of the minority class. This maximizes overall accuracy at the expense of model's capacity for generalization.

Class-imbalance learning is a paradigm designed to improve classification performance by emphasizing underrepresented and hard-to-classify samples. Common implementation strategies include data sampling, ensemble learning, and cost-sensitive approaches (Zhenxing, Yujuan, & Yun, 2019). Among these, sampling-based methods such as oversampling and undersampling are widely used to rebalance class distributions. However, they often fail to account for the underlying data distribution. Undersampling may lead to the loss of informative majority samples, while oversampling can increase the risk of overfitting. Ensemble learning addresses the imbalance by combining multiple complementary classifiers to improve robustness and generalization (Na, Xiaomei, Ershi, Man, Ling, & Bo, 2020; Ribeiro & Reynoso-Meza, 2020). Cost-sensitive learning assigns higher misclassification penalties to minority class instances to guide the model towards balanced decision boundaries, but it is based heavily on carefully chosen cost values, which are often difficult to estimate in practice (Zhou & Zhou, 2016).

Despite the progress made by existing methods in addressing class imbalance issues, they often overlook the unique intrinsic features of time series data and the complex high-order relationships between series. Time series often exhibit not only temporal dependency but also inter-series similarities and differences that reflect shared patterns or discriminative features. Isolated sample-based methods often treat each time series as an independent sample point, failing to fully leverage the associations between series to strengthen the model's recognition and understanding of minority class. Capturing these high-order relationships is especially beneficial for the minority class recognition, as it allows the model to infer class-relevant signals from contextually similar series. However, due to the high dimensionality and variability

* Corresponding author.
E-mail addresses: chang.liu@nuaa.edu.cn (C. Liu), dhguan@nuaa.edu.cn (D. Guan), yuanweiwei@nuaa.edu.cn (W. Yuan), cetinkoc@ucsb.edu (Ç.K. Koç).

https://doi.org/10.1016/j.neunet.2025.107770
Received 15 October 2024; Received in revised form 11 June 2025; Accepted 16 June 2025
Available online 28 June 2025

of time series data, how to effectively represent and utilize these relationships to enhance the capture of the minority class information is an extremely challenging problem.

In response to the aforementioned issues, we utilize graph theory and propose a novel graph-based imbalanced time series classification method called ITS2Graph, aimed at addressing the limitations of traditional methods in handling imbalanced datasets. We begin by encoding time series data using an auto-encoder to extract its latent low-dimensional representation, capturing the key features of the data. Then, we transform each time series into nodes in a graph, using the extracted latent representation as the attributes of the nodes in the graph, thereby converting the time series classification problem into a graph node classification problem. In this manner, we can utilize the similarity of node attributes to define the connectivity between nodes, thus addressing the time series classification problem within the framework of a graph and capturing high-order associations between series. To tackle the class imbalance issue, we introduce a graph generator to generate synthetic attributes and network topological structures for minority class nodes, thereby increasing the number of minority class nodes in the graph, balancing the distribution of nodes across different classes, and enhancing the model's ability to capture information about minority class. Finally, we train a Graph Convolutional Network (GCN) as a discriminator to distinguish between real nodes and fake nodes introduced by the graph generator, as well as between minority and majority class nodes. In this way, compared to existing methods, the proposed ITS2Graph is better equipped to capture high-order intrinsic associations and latent patterns between time series. It addresses class imbalance issues more effectively, achieving superior performance in classification tasks.

The main contributions of this paper are summarized as follows:

- We propose a novel graph-based framework for imbalanced time series classification, which transforms time series into graph-structured data and reformulates the task as a node classification problem. By constructing a similarity-based graph from latent representations extracted by an auto-encoder, the method effectively models high-order relationships among time series and enhances the identification of minority class instances.
- We employ a graph generator to generate synthetic attributes and network topological structures for the minority class nodes and utilize a GCN as a discriminator for adversarial training. We generate more realistic minority class instances by aggregating features from neighboring nodes. This effectively addresses the issue of class imbalance.
- We conducted experiments on 21 UCR datasets and a large-scale medical time series dataset, comparing ITS2Graph with 11 advanced baselines. The results demonstrate that ITS2Graph can effectively handle imbalanced time series data, outperforming baseline methods in terms of AUC, F1 and G-mean, three imbalanced classification metrics.

2. Related work

In this section, we review three research directions relevant to our study: (1) traditional methods for imbalanced time series classification, (2) graph-based approaches for time series analysis, and (3) generative adversarial networks for handling class imbalance.

2.1. Imbalanced Time Series Classification

Class imbalance in Time Series Classification has been extensively studied. Sampling methods address class imbalance by either generating additional minority samples (oversampling) or removing majority samples (undersampling) (Li et al., 2024). SMOTE (Chawla N. V, 2002) synthesizes new samples by interpolating between existing ones and their neighbors. Its extensions, K-means SMOTE (Douzas, Bacao, & Last, 2018) and MWMOTE (Barua, Islam, Yao, & Murase, 2014), enhance diversity or apply weights based on proximity to the majority class, while OREM (Zhu, Liu, & Zhu, 2023) focuses on generating samples in cleaner regions without using k-NN or clustering. However, these approaches often overlook the temporal dependencies in time series data. T-SMOTE (Zhao et al., 2022) improves this by generating samples near class boundaries while preserving sequential patterns. Despite such efforts, oversampling for time series remains computationally expensive and susceptible to overfitting due to unrealistic assumptions and limited variability (He, Bai, Garcia, & Li, 2008). In contrast, undersampling can lead to the loss of valuable majority-class information (XuYing, Jianxin, & ZhiHua, 2009).

Ensemble learning methods combine multiple classifiers to improve robustness (Hastie & Trevor, 2008). SMOTEBoost (Chawla, Lazarevic, Hall, & Bowyer, 2003) integrates SMOTE with boosting to enhance minority class prediction, while RUSBoost (Seiffert, Khoshgoftaar, Van Hulse, & Napolitano, 2009) employs random undersampling to balance classes before applying boosting. CUSBoost (Rayhan et al., 2017) employs clustering-based undersampling to preserve representative majority class samples. imDEF (Zhu et al., 2025) improves the recognition of minority and hard-to-classify samples in imbalanced data by generating synthetic datasets and dynamically selecting the most competent subset of classifiers through multiple rounds. Although these methods are effective to some extent, they often overlook the structural dependencies inherent in time series data.

Cost-sensitive learning mitigates imbalance by assigning higher misclassification costs to the minority class (Castro & Braga, 2013; Krawczyk, 2016a). ACS-ATCN (Zhang, Peng, Zhang, & Wang, 2023) combines attention-based temporal convolutional networks with adaptive top-k differential evolution to jointly optimize costs and hyperparameters, achieving higher G-mean with lower computational cost than traditional oversampling. AdaCC (Iosifidis, Papadopoulos, Rosenhahn, & Ntoutsi, 2023) dynamically adjusts costs during boosting based on cumulative errors, avoiding the need for predefined cost matrices. However, selecting appropriate cost weights remains challenging, especially for high-dimensional time series with complex temporal patterns.

Novel loss function approaches have emerged to address class imbalance more effectively. Ircio, Lojo, Mori, Malinowski, and Lozano (2023) proposed a minimum recall-based loss that prioritizes the lowest class recall, boosting minority recall with minimal accuracy trade-off. Similarly, Chaomin, Xu, and Hao (2025) introduced the COCL loss, which enhances class separability by reducing inter-class similarity and enhancing intra-class compactness.

2.2. Graph-based methods for Time Series Analysis

Graph-based approaches have gained attention for modeling relationships between time series. SimTSC (Zha, Lai, Zhou, & Hu, 2022) transforms time series classification into node classification by constructing a graph based on dynamic time warping (DTW) similarity measures. Similarly, Time2Graph+ (Cheng et al., 2023) maps time series to shape evolution graphs and leverages graph attention networks to learn dynamic and seasonal patterns. Recent advances in graph-based time series classification have shown promising results. VGbel (Wu, Liang, Wang, & Chen, 2023) introduces an ensemble learning framework based on visibility graphs. It employs directed series-to-graph transformation to preserve temporal information and extract multiscale features.

These methods demonstrate the potential of graph structures for time series classification, but often assume balanced class distributions, limiting their applicability to imbalanced scenarios. The ITS2Graph framework builds on these ideas by integrating graph construction with generative adversarial learning to specifically address class imbalance.

2.3. Generative adversarial networks for imbalanced data

Generative Adversarial Networks (GANs) have become a powerful tool for addressing class imbalance in time series classification by generating synthetic minority class samples. Initially introduced by Goodfellow et al. (2014), GANs involve a generator creating synthetic data and a discriminator evaluating its authenticity, a concept extended to time series with methods like TimeGAN (Yoon, Jarrett, & Van der Schaar, 2019), which models temporal dynamics, and C-RNN-GAN (Mogren, 2016), which uses RNNs for sequential data generation. While these approaches excel in producing realistic time series, they often focus on general data augmentation rather than directly tackling class imbalance.

More targeted solutions, such as IB-GAN (Deng, Han, Dreossi, Lee, & Matteson, 2022), adopt a unified framework that combines data imputation and augmentation to generate high-quality synthetic samples for minority class, while graph-based GANs like ImGAGN (Qu, Zhu, & Zheng, 2020) leverage structural relationships to enhance sample quality. Despite advances, issues like capturing high-order correlations and handling high-dimensional data remain. Our model builds on these ideas, using an auto-encoder to map time series to graphs and a graph generator with a GCN discriminator to balance classes and capture complex associations, offering a novel approach to imbalanced time series classification.

3. Proposed method

Algorithm 1 ITS2Graph for imbalanced time series classification

Input: Time series $\mathcal{X} = [X_1, X_2, \ldots, X_n]$
Graph Construction:
1: **for** each epoch **do**
2: **for** $X \leftarrow \mathcal{X}$ **do**
3: Calculate the encoding H with Eq. (1).
4: Calculate the reconstruction $\hat{X}$ with Eq. (2).
5: **end for**
6: Calculate the loss $\mathcal{L}_R$ with Eq. (3) and minimize it with Eq. (4).
7: **end for**
8: Compute the similarity matrix S for $\mathcal{H}$ with Eq. (5).
9: Obtain the unbalanced graph network G_{imb}.
Graph-Based Generative Adversarial Learning:
10: **for** each epoch **do**
11: Sample a batch of noise samples Z from noise prior $p_z(z)$.
12: Generate synthetic minority nodes feature and their connections with Eq. (6) and Eq. (7).
13: Obtain a new balanced graph G_b.
14: **for** λ steps **do**
15: Calculate the loss L_{dis} with Eq. (10).
16: Perform the gradient descent step on the discriminator to minimize L_{dis}.
17: **end for**
18: Calculate the loss L_{gen} with Eq. (9).
19: Perform the gradient descent step on the generator to minimize L_{gen}.
20: **end for**

To address the issues in traditional classification methods for imbalanced time series, we propose a novel graph-based solution framework called ITS2Graph.

Fig. 1 shows an overview of ITS2Graph. ITS2Graph mainly consists of two parts: (1) the Graph Construction and (2) the Graph-Based Generative Adversarial Learning. Algorithm 1 summarizes the process of using ITS2Graph for imbalanced time series classification.

3.1. Graph construction

This section describes how to construct a graph based on the similarities of the latent representations extracted by the Auto-encoder (AE).

3.1.1. Auto-encoder for representation learning

Considering the original time series data may have high dimensionality, we employ the AE to extract a latent low-dimensional representation. This not only reduces computational complexity but also eliminates noise and redundant information.

Let $\mathcal{X} = [X_1, X_2, \ldots, X_n]$ represent the set of all time series. The AE first encodes the series $\mathcal{X}$ through a linear mapping and a nonlinear activation function:

$$H_i = g\left(W X_i + b_m\right), \tag{1}$$

where $X_i \in X$, W is the weight matrix between the encoding layer and the input layer, b_m is the bias vector of the encoding layer nodes, and $g(\cdot)$ is the activation function of the nodes.

The decoder completes the decoding of the encoded features to obtain the reconstruction $\hat{\mathcal{X}}$ of the input samples. The decoding process is similar to the encoding process:

$$\hat{X}_i = g\left(W^T H_i + b_d\right), \tag{2}$$

where b_d is the bias vector of the decoding layer.

The squared error is taken as the loss function. For the input sample $\mathcal{X} = [X_1, X_2, \ldots, X_n]$ and the reconstruction $\hat{\mathcal{X}} = [\hat{X}_1, \hat{X}_2, \ldots, \hat{X}_n]$, the loss function is given by Eq. (3):

$$\mathcal{L}_R = \frac{1}{2}\sum_{i=1}^{n}\left\|\hat{X}_i - X_i\right\|_2^2. \tag{3}$$

Train the AE to minimize the loss function:

$$\arg\min_{W,b} \mathcal{L}(W, b). \tag{4}$$

Adopt the encoding obtained after the final iteration as the latent representation $\mathcal{H} = [H_1, H_2, \ldots, H_n]$ of the original time series.

3.1.2. Graph construction with representation similarity

After obtaining the latent representation $\mathcal{H} = [H_1, H_2, \ldots, H_n]$ of the original time series, we calculate the similarity between each representation vector. Here, we use cosine similarity to compute the similarity matrix S:

$$S = \begin{bmatrix} \cos(H_1, H_1) & \cos(H_1, H_2) & \cdots & \cos(H_1, H_n) \\ \cos(H_2, H_1) & \cos(H_2, H_2) & \cdots & \cos(H_2, H_n) \\ \vdots & \vdots & \ddots & \vdots \\ \cos(H_n, H_1) & \cos(H_n, H_2) & \cdots & \cos(H_n, H_n) \end{bmatrix}, \tag{5}$$

where $\cos(H_i, H_j) = \frac{H_i \cdot H_j}{\|H_i\|\|H_j\|}$, and $\|\cdot\|$ is the Euclidean norm of the vector. Here, S is a symmetric matrix.

Given the similarity matrix S, we construct the graph as described below. To filter out irrelevant neighbors, we retain only the top k neighbors for each node. Specifically, for each row in S with N entries, we keep only the top k entries with the highest values, setting them to 1, and set all other entries to 0. This results in a binary sparse matrix A, which is then used as the adjacency matrix of the nodes in the graph.

By this approach, we transform the imbalanced time series $\mathcal{X} = [X_1, X_2, \ldots, X_n]$ into graph data, obtaining the imbalanced graph network $G_{\text{imb}} = (V, E, A, H, C)$, where $V = [1, 2, \ldots, n]$ is the set of nodes, E is the set of edges, A is the adjacency matrix, H is the node feature matrix with latent representations extracted by the AE, and C is the set of node classes. In the imbalanced graph network G_{imb}, $|C_{min}|$ is much smaller than $|C_{maj}|$, where C_{min} and C_{maj} denote the sets of nodes for the minority and majority classes, respectively.

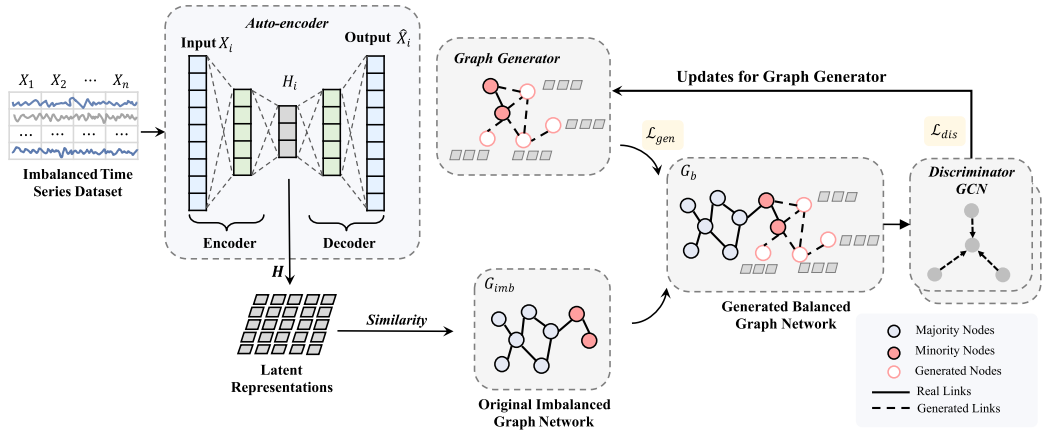


Fig. 1. An overview of the proposed ITS2Graph framework. Each time series corresponds to a node in the graph, and the pairwise similarities of the representations extracted by AE correspond to the links between nodes, resulting in an imbalanced graph network. A graph generator is utilized to construct a balanced graph network, with a GCN serving as a discriminator for adversarial training.

3.2. Graph-based generative adversarial learning

In Section 3.1, we transformed time series into graph data and established high-order connections between series. To address the class imbalance issue of graph nodes, in this section, we propose a method based on GANs.

3.2.1. Graph generator

The generator for graph data has to perform a more complicated task than others deal with typical Euclidean space data. It must simultaneously grasp the distribution of node attribute features and the topological structure distribution of connections between nodes.

In this work, the graph generator is a fully connected neural network that receives noise vectors as input and outputs the topology and features of the graph, denoted as $G_{\text{graph}} : \mathcal{Z} \to \mathcal{H} \times \mathcal{T}$. Here, $\mathcal{Z}$ is the noise space, and $\mathcal{H}, \mathcal{T}$ are the graph network's feature and structure space, respectively.

In particular, for the imbalanced graph network $G_{\text{imb}} = (V, E, A, H, C)$, let n_{maj} and n_{min} denote the number of majority and minority nodes, respectively, with $n = n_{\text{maj}} + n_{\text{min}}$. Let n_g denote the number of nodes that need to be generated, $n_g = n_{\text{maj}} - n_{\text{min}}$, thereby obtaining a graph with the balanced distribution of node classes. Therefore, the number of units in the input layer is d_Z, which is the noise space dimension. The number of units in the output layer is $d_o = n_g \times n_{\text{min}}$. Then, transforming the output vector $\mathbf{o} \in \mathbb{R}^{d_o}$ into matrix form $O \in \mathbb{R}^{n_g \times n_{\text{min}}}$, and normalizing each row in O by using the $softmax(\cdot)$ function as Eq. (6):

$$T_i = \text{softmax}\left(O_i\right) = \frac{e^{O_{ij}}}{\sum_{i=1}^{n_g}\sum_{j=1}^{n_{\text{min}}} e^{O_{ij}}}, \tag{6}$$

where each row O_i of the matrix O denotes the connections between each generated minority node and all original real minority nodes. Meanwhile, each element T_{ij} reflects the normalized connection weight between the original minority node $v_j \in V$ and the generate node $u_i \in U$, where U is the set of generated minority nodes. Thus, the information about the network topological structure between the original minority nodes and the generated minority nodes is represented by T.

The attribute features of the generated minority nodes, $H_g \in \mathbb{R}^{n_g \times f}$, are obtained by aggregating the attribute features of the real minority nodes that are linked to them, which also allows us to generate more realistic minority nodes. We use Eq. (7) to aggregate the neighbor node attributes of each generated minority node:

$$H_g = T H_{\text{min}}, \tag{7}$$

where $H_{\text{min}} \in \mathbb{R}^{n_{\text{min}} \times f} \subset H$ is the feature matrix of the real minority nodes from the original imbalanced graph G_{imb}, and f is the dimension of the original features.

3.2.2. Discriminator

For the discriminator, we use a GCN. The input to the GCN is a new graph $G_b = \left(V', E', A', H', C'\right)$, where has a balanced node class distribution, obtained by incorporating the minority nodes generated from the graph generator into the original imbalanced graph G_{imb}. In this case, V' denotes the new set of nodes consisting of the minority class nodes generated by graph generator and the original nodes from G_{imb}. E' represents the new set of edges comprising the edges generated by graph generator and original edges from G_{imb}. A' is the new adjacency matrix related to V'. H' is the new feature matrix. C' represents the new node label set, which formed by combining the attributes 'real' or 'fake' with 'minority' or 'majority', i.e., $C' = \{(real, minority), (real, majority), (fake, minority)\}$. Because the graph generator does not generate majority nodes, the set C' does not contain the label $(fake, majority)$.

The objective of the discriminator is to distinguish between real nodes and fake nodes generated by the Graph Generator on the balanced graph network, as well as to differentiate between minority and majority class nodes. Hence, the GCN can be utilized as a multi-class node classifier. The output of the GCN is computed by the $softmax(\cdot)$ function as Eq. (8):

$$Y = \text{softmax}\left(\hat{A}' \,\text{ReLU}\left(\hat{A}' H'^{\Omega^0}\right)\Omega^1\right), \tag{8}$$

where Ω^0 and Ω^1 are input-to-hidden and hidden-to-output weight matrices, respectively.

3.2.3. Optimization

The loss function of graph generator consists of four components as Eq. (9). The first term $\mathcal{L}_{rf}$ and the second term $\mathcal{L}_{mi}$ are the confusing discriminator loss on the generated minority data. The third term $\mathcal{L}_{di}$ is intended to bring the generated minority nodes closer to the real minority nodes, and the last term $\mathcal{L}_{re}$ is the regularizer.

$$\begin{aligned}\mathcal{L}_{\text{gen}} &= \mathcal{L}_{rf} + \mathcal{L}_{mi} + \mathcal{L}_{di} + \mathcal{L}_{re} \\ &= \sum_{i=1}^{n_g} -q_i \log \Pr\left(\hat{y}_i = \text{real} \mid \mathbf{h_i}\right) \\ &\quad + \sum_{i=1}^{n_g} -q_i \log \Pr\left(\hat{y}_i = \text{minority} \mid \mathbf{h_i}\right) \\ &\quad + \frac{1}{\left|n_g\right|} \sum_{i=1}^{n_g}\sum_{j=1}^{n_{\text{min}}} \|\mathbf{h_i} - \mathbf{h_j}\|_2^2 \\ &\quad + \alpha\|\Theta\|_2^2,\end{aligned} \tag{9}$$

where $q_i \in C'$ represents the true labels, and $\hat{y}_i \in Y$ represents the output of the discriminator, which is the prediction probability. $\mathbf{h_i}$ is

the embedding vector of the node. Θ is the set of training weights of graph generator with regularization coefficient α.

The loss function of discriminator consists of four components as Eq. (10). The first term $\mathcal{L}_{fa}$ is the cross entropy loss to distinguish between the node generated by graph generator and the real node of the original network. The second term $\mathcal{L}_{cl}$ is the cross entropy loss to distinguish between the node of minority class and the node of majority class. The third term $\mathcal{L}_{mm}$ is designed to ensure that the embeddings of nodes from different classes are sufficiently distant from each other. The last term $\mathcal{L}_{ree}$ is regularizer.

$$\begin{aligned}\mathcal{L}_{\text{dis}} &= \mathcal{L}_{fa} + \mathcal{L}_{cl} + \mathcal{L}_{mm} + \mathcal{L}_{ree} \\ &= \sum_{i=1}^{n_g+n_{\min}+n_{\text{maj}}} -\Big[q_i \log\left(\Pr\left(\hat{y}_i = \text{fake} \mid \mathbf{h_i}\right)\right) \\ &\quad + \left(1-q_i\right) \log\left(1 - \Pr\left(\hat{y}_i = \text{fake} \mid \mathbf{h_i}\right)\right)\Big] \\ &\quad + \sum_{i=1}^{n_g+n_{\min}+n_{\text{maj}}} -\Big[q_i \log\left(\Pr\left(\hat{y}_i = \text{minority} \mid \mathbf{h_i}\right)\right) \\ &\quad + \left(1-q_i\right) \log\left(1 - \Pr\left(\hat{y}_i = \text{minority} \mid \mathbf{h_i}\right)\right)\Big] \\ &\quad - \sum_{i=1}^{n_{\min}} \sum_{j=1}^{n_{\text{maj}}} \left\|\mathbf{h_i} - \mathbf{h_j}\right\|_2^2 \\ &\quad + \beta \|\Omega\|_2^2, \end{aligned} \tag{10}$$

where Ω is the set of training weights of the discriminator with regularization coefficient β.

The objective function for adversarial training in graph-based generative adversarial learning is as Eq. (11):

$$\begin{aligned}\min_G \max_D V(D,G) = {} & E_{x\sim p_{\text{data}}(x)}\left[\log D(x) + \mathcal{L}_{cl} + \mathcal{L}_{mm} + \mathcal{L}_{ree}\right] \\ & + E_{z\sim p_z(z)}\left[\log(1 - D(G(Z))) + \mathcal{L}_{mi} + \mathcal{L}_{di} + \mathcal{L}_{re}\right]. \end{aligned} \tag{11}$$

where G represents the graph generator, and D represents the discriminator.

Graph generator aims to generate synthetic minority class nodes that approximate the data distribution features of real minority class nodes, with the purpose of introducing confusion during model training and enhancing the discriminator's ability to differentiate. The core objective of the graph-based generative adversarial learning is to accurately distinguish between nodes generated by graph generator and real nodes in the original data, and correctly identify minority class nodes and majority class nodes within the graph. This optimizes the model's handling efficiency for imbalanced datasets.

4. Experiment

To evaluate the effectiveness and generalization ability of the proposed ITS2Graph framework, we conduct comprehensive experiments on 21 benchmark UCR time series datasets and a large-scale real-world medical time series dataset. We compare ITS2Graph with a variety of state-of-the-art baselines and analyze its performance under different structural and training configurations.

4.1. Datasets

We evaluate our method on 21 benchmark datasets from the UCR Time Series Archive, which are widely adopted in time series classification research and span a broad range of sequence lengths, class imbalance ratios, and sample sizes.

For all datasets, following Cao, Li, Woon, and Ng (2013) and Gong and Chen (2016): since these datasets originally include multiple classes, we transform them into binary classification by selecting one class as the positive class and considering the remaining classes as the negative class. To verify the broad applicability of our proposed method, we select datasets with imbalance ratios $r \in [1.82, 30.00]$, covering both low and high levels of imbalance. The summary of the datasets in our experiment is shown in Table 1.

To further assess the scalability and practical relevance of our model, we additionally include the MIT-BIH Arrhythmia Dataset from PhysioNet, a large-scale real-world dataset comprising multichannel electrocardiogram (ECG) recordings for arrhythmia detection. The heartbeats are categorized into two classes: normal rhythms and abnormal rhythms (i.e., arrhythmias), with an initial imbalance ratio of 2.18. To further evaluate the effectiveness of the model under more challenging imbalanced conditions, we apply undersampling to increase the level of imbalance. The resulting dataset contains 115,505 samples, with an adjusted imbalance ratio of approximately 5.39.

4.2. Compared methods

To validate the effectiveness of the proposed method ITS2Graph, we compare it with the following 11 advanced methods, which can be divided into two classes: imbalanced time series classification methods (i.e., SMOTE, K-means SMOTE, MWMOTE, T-SMOTE, SMOTEBoost, RUSBoost, CUSBoost, MR, and COCL) and balanced time series classification methods (i.e., simTSC, Time2Graph+). The imbalanced methods encompass four major types: sampling-based, ensemble-based, cost-sensitive, and loss function-based approaches. All comparison methods are introduced as follows:

- SMOTE (Chawla N. V, 2002): A classical method that generates synthetic minority class samples by interpolating between existing samples and their k-nearest neighbors.
- K-means SMOTE (Douzas et al., 2018): Enhances SMOTE by clustering minority samples and interpolating within each cluster for better diversity.
- MWMOTE (Barua et al., 2014): Selects hard-to-learn minority instances and assigns weights based on their proximity to majority class samples to guide the oversampling.
- T-SMOTE (Zhao et al., 2022): A temporal-oriented oversampling method that generates samples near class boundaries while preserving temporal dynamics, especially effective for early prediction tasks.
- SMOTEBoost (Chawla et al., 2003): Combines SMOTE with AdaBoost to iteratively improve minority class performance.
- CUSBoost (Rayhan et al., 2017): Applies k-means clustering for representative undersampling, followed by boosting.
- RUSBoost (Seiffert et al., 2009): Integrates random undersampling with boosting to mitigate imbalance.
- MR (Ircio et al., 2023): A smooth approximation to the minimum recall objective that encourages balanced recall across classes, especially improving performance on minority classes.
- COCL (Chaomin et al., 2025): A contrastive clustering loss that enhances class separability by minimizing intra-class distance and maximizing inter-class distance, tailored for imbalanced time series classification.
- simTSC (Zha et al., 2022): Converts time series classification into a graph node classification problem using DTW-based similarity measures.
- Time2Graph+ (Cheng et al., 2023): Maps time series into shape evolution graphs and learns temporal dynamics using graph attention mechanisms.

4.3. Implementation details

All the experiments are implemented in PyTorch 1.13.1 and conducted on NVIDIA Tesla A100 80 GB GPU.

The hyperparameters for the ITS2Graph algorithm are set as follows. In the graph construction part, we set the number of neighbors $k = 4$ for each node. In the part of graph-based generative adversarial learning, for the generator, it consists of 3 fully connected layers: an input layer

Table 1
Summary of dataset characteristics.

Dataset	#Sample	Positive class	Imbalance ratio	Time series length
Adiac	390	25	30.00	176
Car	60	3	4.45	577
CricketX	390	3	10.47	300
CricketY	390	3	9.83	300
CricketZ	390	1	11.19	300
Crop	7200	24	23.00	46
ECG5000 (ECG)	500	2	1.82	140
FaceAll	560	14	13.00	131
Fish	175	3	5.25	463
Haptics	155	2	3.56	1092
InsectWingbeatSound (IWS)	220	1	10.00	256
MixedShapesRegularTrain (RTrain)	500	5	4.00	2425
MixedShapesSmallTrain (STrain)	100	5	4.00	2425
Plane	105	7	4.25	144
Trace	100	4	3.00	275
UWaveGestureLibraryAll (UWaveAll)	896	8	7.96	945
UWaveGestureLibraryX (UWaveX)	896	2	7.30	315
UWaveGestureLibraryY (UWaveY)	896	2	7.30	315
UWaveGestureLibraryZ (UWaveZ)	896	2	7.30	315
Wafer	1000	−1	9.31	152
Worms	181	2	4.84	900

Table 2
The comparative results with respect to AUC.

Dataset	Sim TSC	Time2 Graph+	SMOTE	MW-MOTE	K-means SMOTE	T-SMOTE	SMOTE Boost	RUS Boost	CUS Boost	MR	COCL	ITS2Graph
Adiac	0.6342	0.6098	0.6437	0.6635	0.6491	0.7524	0.7022	0.6939	0.6823	0.7576	0.7164	**0.7613**
Car	0.5730	0.5793	0.7673	0.7693	0.7785	0.7899	0.7792	0.7843	0.7802	0.7824	0.7908	**0.8008**
CricketX	0.6293	0.6485	0.9776	0.9721	0.9892	0.9917	0.9843	0.9795	0.9835	0.9912	0.9906	**0.9940**
CricketY	0.6591	0.6556	0.8934	0.9132	0.9043	0.9202	0.9129	0.9245	0.9176	0.9454	0.9515	**0.9565**
CricketZ	0.6354	0.6823	0.8904	0.8921	0.9094	0.9148	0.8917	0.8935	0.8929	0.9309	0.9401	**0.9415**
Crop	0.5730	0.5793	0.9673	0.9693	0.9785	0.9889	0.9886	**0.9893**	0.9539	0.9703	0.9890	**0.9893**
ECG	0.5904	0.6007	0.8791	0.8492	0.8505	0.9765	0.9374	0.9755	0.9729	0.9632	0.9791	**0.9809**
FaceAll	0.6392	0.6122	0.9432	0.9211	0.9527	0.9481	0.8669	0.9536	0.9437	0.9500	0.9546	**0.9585**
Fish	0.6254	0.6917	0.8987	0.8935	0.9021	0.9375	0.9076	0.9156	0.9198	0.9384	0.9414	**0.9471**
Haptics	0.5021	0.4756	0.7215	0.7243	0.7481	0.7558	0.7489	0.7432	0.7587	0.7654	0.7657	**0.7725**
IWS	0.6742	0.6693	0.9382	0.9230	0.9480	0.8134	0.8608	0.9529	0.9585	0.9535	0.9501	**0.9682**
RTrain	0.6242	0.6143	0.9689	0.9792	0.9713	0.9180	0.9866	0.9874	0.9769	0.9744	0.9886	**0.9892**
STrain	0.6194	0.6005	0.9692	0.9588	0.9725	0.9599	0.9213	0.9870	0.9873	0.9464	0.9877	**0.9908**
Plane	0.7212	0.7367	0.9543	0.9632	0.9789	0.9907	0.9734	0.9872	0.9865	0.9959	0.9972	**0.9974**
Trace	0.5039	0.5190	0.8762	0.8603	0.8514	0.9194	0.8021	0.8793	0.8860	0.9178	**0.9203**	0.9179
UWaveAll	0.6210	0.6108	0.9217	0.9389	0.9575	0.9449	0.9427	0.9698	0.9531	0.9592	0.9787	**0.9954**
UWaveX	0.5893	0.5932	0.9381	0.9531	0.9445	0.9584	0.9363	0.9582	0.9487	0.9552	0.9842	**0.9902**
UWaveY	0.5995	0.6073	0.9206	0.9298	0.9564	0.9709	0.9123	0.9805	0.9484	0.9752	0.9699	**0.9918**
UWaveZ	0.5884	0.5919	0.9320	0.9209	0.9481	0.9747	0.9517	0.9723	0.9301	0.9726	0.9741	**0.9787**
Wafer	0.6521	0.6392	0.9495	0.9374	0.9561	0.9807	0.9565	0.9909	0.9853	0.9806	0.9913	**0.9944**
Worms	0.6345	0.6289	0.9623	0.9765	0.9782	0.9899	0.9732	0.9742	0.9643	0.9940	0.9958	**0.9964**
MIT-BIH	0.6845	0.6521	0.9456	0.9487	0.9634	0.9901	0.9745	0.9783	0.9754	0.9869	0.9802	**0.9915**
Average	0.6170	0.6181	0.9027	0.9026	0.9131	0.9267	0.9050	0.9305	0.9230	0.9367	0.9426	**0.9502**

with 100 units and a hidden layer with 200 units, the number of units in the output layer is equal to the difference between the majority and minority classes in the training set. For the discriminator, it comprises two layers of GCN and a softmax function. Additionally, we update the generator and discriminator at a ratio of 1:50. The learning rate is set to be 0.0001.

We employ three common classification metrics to assess the performance of all algorithms: (1) F1, which is the harmonic mean of precision and recall, measuring the model's comprehensive performance in minority class classification tasks; (2) Area Under Curve (AUC), which measures the model's ability to correctly distinguish between positive and negative classes at all possible classification thresholds; (3) G-mean: which evaluates the balanced recognition ability of classifiers on both positive and negative classes in imbalanced datasets, is defined as the geometric mean of sensitivity (recall for the positive class) and specificity (recall for the negative class). We implement the baselines using their official code and adhere to the configurations detailed in their respective original papers. Each experiment is run five times, and we report the average results.

4.4. Results

To evaluate the performance of ITS2Graph, we follow the standard linear baseline evaluation scheme. Tables 2, 3 and 4 show the comparative results of ITS2Graph and its 11 baseline methods on 22 datasets. For each dataset, the best results among all 12 methods are highlighted in bold, while the second best results are indicated by underlining. ITS2Graph achieves the best performance on the vast majority of datasets and obtains the best average results across all datasets.

From experimental results, we have a couple of observations.

- Overall, compared to all methods, ITS2Graph achieved the best performance on the vast majority of datasets in terms of AUC, F1, and G-mean metrics. Its average AUC, F1, and G-mean values were 0.9502, 0.8124, and 0.9119, respectively, surpassing the best baseline method by 0.80%, 1.60%, and 1.61%, respectively. These results indicate that the proposed ITS2Graph is a highly competitive algorithm for effectively handling time series data with class imbalance.

Table 3
The comparative results with respect to F1.

Dataset	Sim TSC	Time2 Graph+	SMOTE	MW-MOTE	K-means SMOTE	T-SMOTE	SMOTE Boost	RUS Boost	CUS Boost	MR	COCL	ITS2Graph
Adiac	0.4334	0.4182	0.5483	0.4939	0.5541	0.5801	0.5772	0.5613	0.5704	0.5674	0.5782	**0.5803**
Car	0.5678	0.4967	0.6974	0.6983	0.7553	0.8159	0.7554	0.7524	0.7812	0.8178	0.7721	**0.8217**
CricketX	0.5685	0.4946	0.7152	0.7398	0.7276	0.8358	0.7842	0.7994	0.8297	0.8538	0.8618	**0.8814**
CricketY	0.5739	0.5423	0.6646	0.6712	0.7087	0.7063	0.6525	0.6459	0.6743	0.6902	0.7129	**0.7341**
CricketZ	0.4946	0.4992	0.6549	0.6639	0.6932	0.6947	0.6721	0.6756	0.6889	0.7012	0.7186	**0.7261**
Crop	0.5430	0.5682	0.7792	0.7629	0.7883	0.7917	0.7702	0.7991	0.7531	0.7844	0.7736	**0.8035**
ECG	0.5783	0.5932	0.6830	0.7021	0.7227	0.9489	0.9003	0.8355	0.9004	0.9417	0.9536	**0.9567**
FaceAll	0.5359	0.5495	0.6905	0.6328	0.6484	0.7916	0.6783	0.7190	0.7072	0.7806	0.7824	**0.7936**
Fish	0.5128	0.4947	0.6753	0.6536	0.6798	0.6901	0.6894	0.6813	0.6571	0.6954	0.6989	**0.7027**
Haptics	0.3245	0.3043	0.4982	0.4971	0.4827	0.5069	0.4916	0.5184	0.5073	0.5811	0.5491	**0.5834**
IWS	0.5983	0.5884	0.6913	0.7107	0.7379	0.7449	0.7532	0.7282	0.7213	0.7452	0.7557	**0.7622**
RTrain	0.4926	0.5102	0.7329	0.7232	0.8509	0.8218	0.7409	0.8311	0.7309	0.8504	**0.8650**	**0.8650**
STrain	0.5307	0.4988	0.7927	0.7892	0.8735	0.8293	0.8417	0.8131	0.6582	0.8609	0.8539	**0.8791**
Plane	0.7039	0.7287	0.9285	0.9372	0.9626	0.9928	0.9754	0.9635	0.9826	0.9914	0.9874	**0.9953**
Trace	0.5097	0.5302	0.6103	0.6483	0.6531	0.6871	0.6122	0.6723	0.6301	0.7192	**0.7317**	0.7140
UWaveAll	0.6018	0.6293	0.7812	0.8043	0.8079	0.8125	0.8609	0.7701	0.7150	0.8478	0.8665	**0.8778**
UWaveX	0.5439	0.5672	0.7719	0.7922	0.7801	0.8665	0.8818	0.7857	0.7228	0.6896	0.8805	**0.8843**
UWaveY	0.5590	0.5832	0.7288	0.7493	0.7530	0.7973	0.8251	0.8272	0.7581	0.7879	0.8254	**0.8322**
UWaveZ	0.4582	0.4903	0.7935	0.7843	0.7743	0.8216	0.8208	0.7705	0.7693	0.7593	0.8217	**0.8238**
Wafer	0.5002	0.5492	0.7451	0.7876	0.7448	0.8955	0.8953	0.8985	0.6733	0.8343	0.9003	**0.9007**
Worms	0.6485	0.6283	0.8124	0.8329	0.7891	0.8743	0.8738	0.8543	0.8841	0.8966	0.8825	**0.9106**
MIT-BIH	0.6198	0.6087	0.6184	0.7093	0.8125	0.8425	0.6547	0.8865	0.6418	0.8302	0.8199	**0.8438**
Average	0.5409	0.5397	0.7097	0.7175	0.7409	0.7886	0.7594	0.7631	0.7253	0.7830	0.7996	**0.8124**

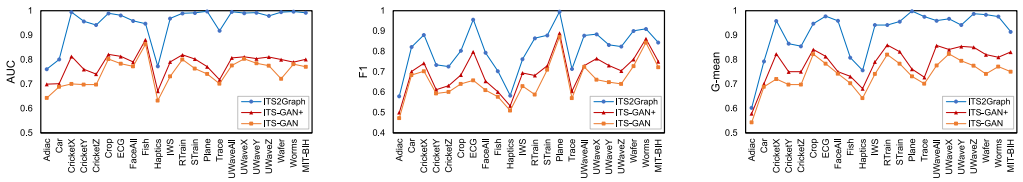
Table 4
The comparative results with respect to G-mean.

Dataset	Sim TSC	Time2 Graph+	SMOTE	MW-MOTE	K-means SMOTE	T-SMOTE	SMOTE Boost	RUS Boost	CUS Boost	MR	COCL	ITS2Graph
Adiac	0.4754	0.4390	0.4845	0.4991	0.4982	0.5383	0.5021	0.5237	0.5283	0.5652	0.6013	**0.6032**
Car	0.5937	0.5642	0.7356	0.7427	0.7489	0.7901	0.7632	0.7767	0.7645	0.7828	0.7734	**0.7929**
CricketX	0.7235	0.6847	0.9036	0.9124	0.8976	0.9557	0.9236	0.9292	0.9143	0.9516	0.9267	**0.9585**
CricketY	0.6145	0.6323	0.7832	0.8056	0.8028	0.8645	0.8121	0.8225	0.8346	0.8536	0.8552	**0.8651**
CricketZ	0.5947	0.6235	0.8026	0.7923	0.8254	0.8455	0.8237	0.8018	0.8354	0.8492	0.8534	**0.8548**
Crop	0.5765	0.5734	0.9043	0.8952	0.8745	0.9392	0.8873	0.8999	0.9052	0.9432	0.9463	**0.9468**
ECG	0.7032	0.6927	0.9224	0.9306	0.9443	0.9748	0.9323	0.9554	0.9434	0.9717	0.9681	**0.9775**
FaceAll	0.7156	0.7023	0.9141	0.9265	0.9232	0.9208	0.9136	0.9224	0.9126	**0.9665**	0.9497	0.9587
Fish	0.5623	0.5932	0.7643	0.7756	0.7732	0.7815	0.7745	0.7763	0.7854	0.7955	0.7723	**0.8081**
Haptics	0.4456	0.4632	0.6923	0.7125	0.7034	0.7513	0.7245	0.7223	0.7332	0.7303	0.7192	**0.7563**
IWS	0.6834	0.6543	0.9024	0.9123	0.9182	0.9286	0.9035	0.9143	0.9136	0.9324	0.9205	**0.9416**
RTrain	0.6245	0.6523	0.8932	0.9056	0.9028	0.9339	0.9134	0.9121	0.9163	0.9243	0.9344	**0.9413**
STrain	0.6667	0.6434	0.9043	0.9152	0.9145	0.9373	0.9294	0.9123	0.9276	0.9441	0.9514	**0.9558**
Plane	0.6595	0.6237	0.9643	0.9727	0.9782	0.9910	0.9719	0.9834	0.9856	0.9887	0.9931	**0.9982**
Trace	0.6342	0.6654	0.9238	0.9343	0.9476	0.9728	0.9421	0.9519	0.9483	0.9713	0.9698	**0.9754**
UWaveAll	0.6123	0.6567	0.8432	0.8789	0.8854	0.8901	0.8922	0.8823	0.9117	0.9244	0.9239	**0.9594**
UWaveX	0.5945	0.6321	0.9032	0.9165	0.9276	0.9429	0.9243	0.9189	0.9223	0.9533	0.9519	**0.9670**
UWaveY	0.6165	0.6087	0.8945	0.8923	0.9054	0.9368	0.9234	0.9189	0.9223	0.9291	0.9398	**0.9416**
UWaveZ	0.6367	0.6789	0.9443	0.9527	0.9545	0.9572	0.9589	0.9507	0.9586	0.9539	0.9776	**0.9873**
Wafer	0.7154	0.6834	0.9423	0.9389	0.9456	0.9737	0.9348	0.9654	0.9539	0.9809	0.9754	**0.9836**
Worms	0.6456	0.6734	0.9367	0.9289	0.9323	0.9713	0.9489	0.9623	0.9543	0.9520	0.9689	**0.9758**
MIT-BIH	0.6543	0.6315	0.8054	0.8234	0.8289	0.8964	0.8123	0.8456	0.8234	0.8656	0.8702	**0.9136**
Average	0.6249	0.6260	0.8529	0.8620	0.8651	0.8952	0.8687	0.8749	0.8770	0.8968	0.8974	**0.9119**

- As expected, the imbalanced time series sampling and classification methods (i.e., SMOTE, K-means SMOTE, MWMOTE, T-SMOTE, SMOTEBoost, RUSBoost, CUSBoost, MR and COCL) perform better than the balanced time series classification methods (i.e., simTSC and Time2Graph+). This highlights the importance of tailored strategies for handling class imbalance, as these methods effectively mitigate the challenges posed by skewed distributions, leading to better recognition of minority class patterns and improved overall classification performance across diverse datasets.
- From an algorithmic perspective, ITS2Graph outperforms these existing sampling-based methods (i.e., SMOTE, K-means SMOTE, MWMOTE, and T-SMOTE), ensemble learning methods (i.e., SMOTEBoost, RUSBoost, and CUSBoost), and loss function-based methods (i.e., MR and COCL) in handling imbalanced data. Based on this, we conclude that, for imbalanced datasets, our graph-based approach effectively captures high-order associations between time series while leveraging the topological structure of the graph to extract richer sample information, generating more realistic minority class instances. This is particularly effective when minority class information is scarce. Compared to several established methods, the proposed ITS2Graph achieves superior results on imbalanced datasets.

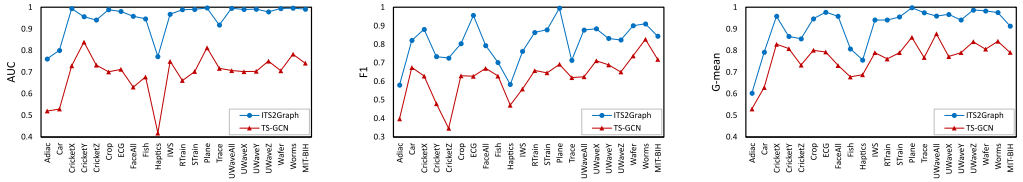
4.5. Influence of graph construction

The primary purpose of graph construction is to capture the high-order intrinsic associations between time series. We utilize an auto-encoder to extract the latent representations of time series and build the graph based on the similarity of each time series representation vector. In this section, we explore the specific impact of graph construction strategies on model performance.



(a) AUC (b) F1 (c) G-mean

Fig. 2. Comparison of the proposed method ITS2Graph and ITS-GAN, ITS-GAN+ on AUC, F1 and G-mean.



(a) AUC (b) F1 (c) G-mean

Fig. 3. Comparison of the proposed method ITS2Graph and TS-GCN on AUC, F1 and G-mean.

To reveal the impact of graph construction, we designed two variants of the proposed method: ITS-GAN and ITS-GAN+. ITS-GAN directly utilizes GANs to generate adversarial imbalanced time series data on the original time series. In contrast, ITS-GAN+ employs the original auto-encoder for feature extraction from the time series and then conducts generative adversarial processing on the extracted latent representation vectors. Fig. 2 shows the comparison results between the proposed ITS2Graph and ITS-GAN, ITS-GAN+ on 22 datasets. Our proposed method achieves better performance than ITS-GAN and ITS-GAN+ in AUC, F1 and G-mean metrics.

The experimental results demonstrate the necessity of the graph construction strategy. Compared to ITS-GAN, which applies GANs directly on the original time series, our proposed ITS2Graph can map time series data into graph structures. This method allows the model to more effectively capture the intrinsic associations and structural information of the time series data, thereby enhancing the ability to capture minority class information in imbalanced data distributions.

Additionally, we also observed the positive impact of feature extraction on model performance. ITS-GAN+ employs the auto-encoder to perform feature extraction and dimensionality reduction on time series, followed by generative adversarial training on the reduced-dimensional latent representations. The performance improvement of ITS-GAN+ over ITS-GAN confirms that dimensionality reduction aids in eliminating noise and redundant information, enabling the model to focus more on the key features in the data.

4.6. Influence of generative adversarial graph network

In the ITS2Graph algorithm, we utilize a graph generator to generate synthetic attributes and network topological structures for the minority class nodes. Thereby increasing the number of minority class nodes in the graph, balancing the distribution of nodes across different classes, and alleviating the class imbalance issue.

To investigate the role of the adversarial generative graph network, we designed a variant called TS-GCN. After graph construction for the time series is complete, we directly employ a GCN as the classifier to replace the generative adversarial module. The experimental comparison between the variant TS-GCN and ITS2Graph on 22 datasets is shown in Fig. 3. The results demonstrate that ITS2Graph outperforms TS-GCN, confirming the critical role of our proposed graph-based adversarial generative learning in addressing the classification of imbalanced time series.

The graph generator increases the number of minority class samples by generating synthetic minority class nodes, which helps the model to better learn and recognize the features of the minority class. Secondly, the graph generator not only generates node attributes, but also learns and generates the topological structure of the graph network, which helps preserve the intrinsic relationships and structural information of the time series data. The ITS2Graph model can more comprehensively understand the complex relationships between time series, further enhancing the accuracy of classification for imbalanced data.

4.7. Visualization of generated nodes

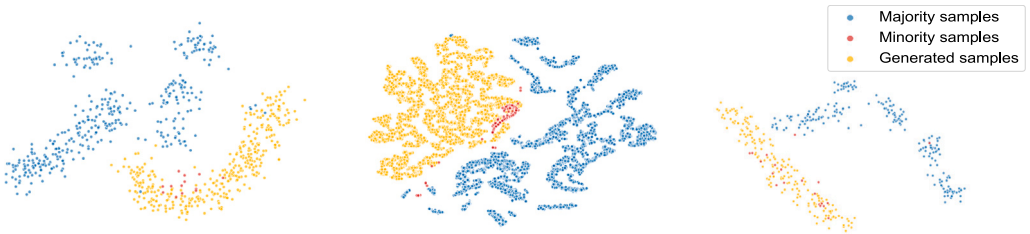
To further investigate the performance of ITS2Graph, we conducted embedding visualizations of real and generated nodes using t-SNE projection across three representative datasets from different domains: Adiac, Crop, and InsectWingbeatSound (IWS). As illustrated in Fig. 4, the generated minority samples (yellow) exhibit a close alignment with the distribution of real minority samples (red), while maintaining a clear distinction from the majority class (blue). This alignment suggests that the graph generator successfully preserves the intrinsic characteristics of the minority class, while the separation from the majority class indicates the introduction of controlled diversity in the synthetic samples.

These visualizations provide strong empirical evidence that the generator not only maintains semantic consistency with the real minority class but also introduces structural diversity, which helps the discriminator learn more discriminative and robust representations. Moreover, the ability of the generator to adaptively model complex and domain-specific feature distributions across diverse datasets validates its effectiveness in capturing high-order temporal and structural associations in imbalanced time series classification tasks.

4.8. Parameter sensitivity analysis

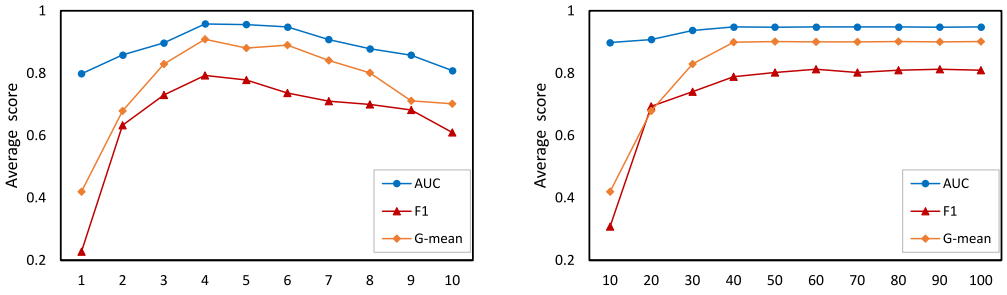
In this section, we analyze the impact of hyper-parameter selection during the experiment on model performance. The experiment involves two main parameters: (1) k, which is the number of neighbors for each node when constructing the graph, and (2) λ, which is the number of discriminator training steps corresponding to each generator training step. We conducted experiments on 22 datasets and reported the performance variation with these hyper-parameters.

The analysis of parameter k essentially examines the impact of graph construction sparsity on model performance. With λ fixed at 50, k was varied from 1 to 10 with an increment of 1. The experimental results are shown in Fig. 5(a), with the best performance achieved



(a) Adiac (b) Crop (c) IWS

Fig. 4. t-SNE visualizations of real and generated samples on three datasets.



(a) k (b) λ

Fig. 5. Hyper-parameter sensitivity analysis of k and λ.

at $k = 4$. When $k = 1$, the graph construction simplifies to the backbone, as there are only self-connections in the graph. As k becomes larger, performance degrades because aggregating more information from neighbors makes the representations more indistinguishable, and an overly dense graph leads to reduced computational efficiency.

Beyond overall performance metrics, the choice of k inherently defines the scope of local structures used to infer higher-order associations between time series. For instance, a very small k (e.g., $k = 1$ or $k = 2$) primarily captures the most immediate strong similarities, potentially overlooking broader relationships within the minority class and leading to a fragmented graph structure. The selected $k = 4$ provides a balance, allowing the model to integrate information from a small neighborhood that is dense enough to reveal localized cluster structures without overly smoothing the representations. As k increases, the graph density increases, incorporating more neighbors. While this can theoretically capture wider associations, it risks diluting the specific characteristics of minority nodes by aggregating less relevant information, making distinct high-order patterns harder to discern, as evidenced by performance degradation at higher k values. The optimal k thus reflects a trade-off between capturing sufficiently complex associations and maintaining discriminative power and efficiency.

In addition to performance trends, we visualized the graph structure under different values of k to observe the impact of graph sparsity. Fig. 6 illustrates the effect of graph sparsity under different values of $k = \{2, 4, 8\}$ by visualizing the 2D t-SNE projection of node embeddings on Wafer dataset. When $k = 2$, the graph becomes overly sparse, resulting in disconnected components and poor minority–majority integration. At $k = 4$, the structure exhibits clearer class boundaries, tighter intra-class cohesion, and well-preserved local neighborhoods, aligning with the best performance observed in AUC, F1 and G-mean metrics. In contrast, $k = 8$ leads to over-smoothing, where excessive neighborhood connections blur class boundaries and reduce structural separability. These observations demonstrate that an appropriate level of sparsity (e.g., $k = 4$) helps maintain high-order associations while preserving meaningful class distinctions in the graph.

For the parameter λ, with k fixed at 4, we varied it from 10 to 100 with an increment of 10. The experimental results are shown in Fig. 5(b). The performance improves with the increase in the number of training steps, and the training effect plateaus when $\lambda = 40$. This is reasonable because the discriminator needs a sufficient number of training steps to learn the node embeddings effectively.

5. Conclusions

In this work, we propose a novel graph-based method to address the issue of class imbalance in time series classification. To capture the high-order associations between time series and the hidden structures of data, especially under the condition of uneven sample distribution, we transform the time series classification problem into a graph node classification problem. Additionally, we introduce a graph generator to generate synthetic attributes and network topological structures for the minority class nodes, thereby balancing the distribution of nodes across different classes. A GCN discriminator is then employed for training. We conducted extensive experiments on 22 real-world time series datasets and compared the proposed ITS2Graph algorithm with various advanced methods. The experimental results demonstrated that ITS2Graph can effectively tackle the problem of imbalanced time series classification.

CRediT authorship contribution statement

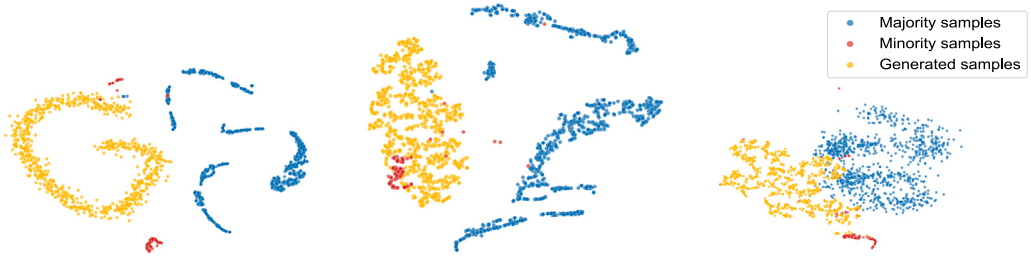
Chang Liu: Writing – review & editing, Writing – original draft, Software, Methodology. **Donghai Guan:** Writing – review & editing, Supervision, Funding acquisition, Conceptualization. **Weiwei Yuan:** Supervision, Formal analysis, Data curation, Conceptualization. **Çetin Kaya Koç:** Investigation, Conceptualization.

Declaration of competing interest

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

Acknowledgments

This work is supported by the National Natural Science Foundation of China (No. 62472220).



(a) $k = 2$ (b) $k = 4$ (c) $k = 8$

Fig. 6. t-SNE visualizations of real and generated samples under different k values on Wafer dataset.

Data availability

Data will be made available on request.

References

Barua, S., Islam, M. M., Yao, X., & Murase, K. (2014). MWMOTE–majority weighted minority oversampling technique for imbalanced data set learning. *IEEE Transactions on Knowledge and Data Engineering*, *26*(2), 405–425.

Cao, H., Li, X. L., Woon, D. Y. K., & Ng, S. K. (2013). Integrated oversampling for imbalanced time series classification. *IEEE Transactions on Knowledge and Data Engineering*, *25*(12), 2809–2822.

Castro, C. L., & Braga, A. P. (2013). Novel cost-sensitive approach to improve the multilayer perceptron performance on imbalanced data. *IEEE Transactions on Neural Networks and Learning Systems*, *24*(6), 888–899.

Chaomin, W., Xu, C., & Hao, W. (2025). A contrastive clustering loss function increases class-balanced in time series classification. *Expert Systems with Applications*, Article 127493.

Chawla, N. V., Lazarevic, A., Hall, L. O., & Bowyer, K. W. (2003). SMOTEBoost: Improving prediction of the minority class in boosting. In *Knowledge discovery in databases* (pp. 107–119). Berlin, Heidelberg: Springer Berlin Heidelberg.

Chawla N. V, H. L. O. (2002). SMOTE: synthetic minority over-sampling technique. *Journal of Artificial Intelligence Research*, *16*, 321–357.

Cheng, Z., Yang, Y., Jiang, S., Hu, W., Ying, Z., Chai, Z., et al. (2023). Time2Graph+: Bridging time series and graph representation learning via multiple attentions. *IEEE Transactions on Knowledge and Data Engineering*, *35*(2), 2078–2090.

Deng, G., Han, C., Dreossi, T., Lee, C., & Matteson, D. S. (2022). Ib-gan: A unified approach for multivariate time series classification under class imbalance. In *Proceedings of the 2022 SIAM international conference on data mining* (pp. 217–225). SIAM.

Douzas, G., Bacao, F., & Last, F. (2018). Improving imbalanced learning through a heuristic oversampling method based on k-means and SMOTE. *Information Sciences*, *465*, 1–20.

Gong, Z., & Chen, H. (2016). Model-based oversampling for imbalanced sequence classification. In *Proceedings of the 25th ACM international on conference on information and knowledge management* (pp. 1009–1018).

Goodfellow, I. J., Pouget-Abadie, J., Mirza, M., Xu, B., Warde-Farley, D., Ozair, S., et al. (2014). Generative adversarial nets. *Advances in Neural Information Processing Systems*, *27*.

Gu, X., Angelov, P. P., & Soares, E. A. (2020). A self-adaptive synthetic oversampling technique for imbalanced classification. *International Journal of Intelligent Systems*, *35*(6), 923–943.

Hastie, & Trevor (2008). *The Elements of Statistical Learning:Data Mining, Inference, and Prediction*. Springer New York.

He, H., Bai, Y., Garcia, E. A., & Li, S. (2008). ADASYN: Adaptive synthetic sampling approach for imbalanced learning. In *2008 IEEE international joint conference on neural networks* (pp. 1322–1328).

He, H., & Garcia, E. A. (2009). Learning from imbalanced data. *IEEE Transactions on Knowledge and Data Engineering*, *21*(9), 1263–1284.

Iosifidis, V., Papadopoulos, S., Rosenhahn, B., & Ntoutsi, E. (2023). AdaCC: cumulative cost-sensitive boosting for imbalanced classification. *Knowledge and Information Systems*, *65*(2), 789–826.

Ircio, J., Lojo, A., Mori, U., Malinowski, S., & Lozano, J. A. (2023). Minimum recall-based loss function for imbalanced time series classification. *IEEE Transactions on Knowledge and Data Engineering*, *35*(10), 10024–10034.

Krawczyk, B. (2016a). Cost-sensitive one-vs-one ensemble for multi-class imbalanced data. In *2016 international joint conference on neural networks* (pp. 2447–2452).

Krawczyk, B. (2016b). Learning from imbalanced data: open challenges and future directions. *Progress in Artificial Intelligence*, *5*(4), 221–232.

Li, S., Song, L., Wu, X., Hu, Z., Cheung, Y.-m., & Yao, X. (2024). Multi-class imbalance classification based on data distribution and adaptive weights. *IEEE Transactions on Knowledge and Data Engineering*.

Mahadevan, S., & Shah, S. L. (2009). Fault detection and diagnosis in process data using one-class support vector machines. *Journal of Process Control*, *19*(10), 1627–1639.

Mogren, O. (2016). C-RNN-GAN: Continuous recurrent neural networks with adversarial training. arXiv preprint arXiv:1611.09904.

Na, L., Xiaomei, L., Ershi, Q., Man, X., Ling, L., & Bo, G. (2020). A novel ensemble learning paradigm for medical diagnosis with imbalanced data. *IEEE ACCESS*, *8*, 171263–171280.

Qu, L., Zhu, H., & Zheng, R. (2020). Imgagn: Imbalanced network embedding via generative adversarial graph networks. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery & data mining* (pp. 1390–1398).

Rayhan, F., Ahmed, S., Mahbub, A., Jani, R., Shatabda, S., & Farid, D. M. (2017). CUSboost: Cluster-based under-sampling with boosting for imbalanced classification. In *2017 2nd international conference on computational systems and information technology for sustainable solution* (pp. 1–5).

Ribeiro, V. H. A., & Reynoso-Meza, G. (2020). Ensemble learning by means of a multi-objective optimization design approach for dealing with imbalanced data sets. *Expert Systems with Applications*, *147*, 113232–113232.

Seiffert, C., Khoshgoftaar, T. M., Van Hulse, J., & Napolitano, A. (2009). RUSboost: A hybrid approach to alleviating class imbalance. *IEEE Transactions on Systems, Man, and Cybernetics-Part A: Systems and Humans*, *40*(1), 185–197.

Waleed, H., Andrew, G. S., & John, Y. (2022). Financial fraud: A review of anomaly detection techniques and recent advances. *Expert Systems with Applications*, *193*.

Woods, K. S., Doss, C. C., Bowyer, K. W., Solka, J. L., Priebe, C. E., & Kegelmeyer, W. P. (1993). Comparative evaluation of pattern recognition techniques for detection of microcalcifications in mammography. *International Journal of Pattern Recognition and Artificial Intelligence*, *07*(06), 1417–1436.

Wu, S., Liang, M., Wang, X., & Chen, Q. (2023). VGbel: An exploration of ensemble learning incorporating non-euclidean structural representation for time series classification. *Expert Systems with Applications*, *224*, Article 119942.

XuYing, L., Jianxin, W., & ZhiHua, Z. (2009). Exploratory undersampling for class-imbalance learning. *IEEE Transactions on Systems, Man and Cybernetics, Part B (Cybernetics)*, *39*(2), 539–550.

Yang, Q., & Wu, X. (2006). 10 challenging roblems in data mining research. *International Journal of Information Technology & Decision Making*, *5*(4), 597–604.

Yoon, J., Jarrett, D., & Van der Schaar, M. (2019). Time-series generative adversarial networks. *Advances in Neural Information Processing Systems*, *32*.

Zha, D., Lai, K. H., Zhou, K., & Hu, X. (2022). Towards similarity-aware time-series classification. In *Proceedings of the 2022 SIAM international conference on data mining* (pp. 199–207).

Zhang, X., Peng, H., Zhang, J., & Wang, Y. (2023). A cost-sensitive attention temporal convolutional network based on adaptive top-k differential evolution for imbalanced time-series classification. *Expert Systems with Applications*, *213*, Article 119073.

Zhao, P., Luo, C., Qiao, B., Wang, L., Rajmohan, S., Lin, Q., et al. (2022). T-SMOTE: Temporal-oriented synthetic minority oversampling technique for imbalanced time series classification. In *IJCAI* (pp. 2406–2412).

Zhenxing, X., Yujuan, F., & Yun, L. (2019). Predictive modeling of the risk of acute kidney injury in critical care: A systematic investigation of the class imbalance problem. *AMIA Joint Summits on Translational Science Proceedings*, *2019*, 809–818.

Zhou, Y. H., & Zhou, Z. H. (2016). Large margin distribution learning with cost interval and unlabeled data. *IEEE Transactions on Knowledge and Data Engineering*, *28*(7), 1749–1763.

Zhu, T., Hu, X., Liu, X., Zhu, E., Zhu, X., & Xu, H. (2025). Dynamic ensemble framework for imbalanced data classification. *IEEE Transactions on Knowledge and Data Engineering*.

Zhu, T., Liu, X., & Zhu, E. (2023). Oversampling with reliably expanding Minority Class Regions for imbalanced data learning. *IEEE Transactions on Knowledge and Data Engineering*, *35*(6), 6167–6181. http://dx.doi.org/10.1109/TKDE.2022.3171706.