Expert Systems With Applications 224 (2023) 119942

Contents lists available at ScienceDirect

Expert Systems With Applications

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

VGbel: An exploration of ensemble learning incorporating non-Euclidean structural representation for time series classification

Shaocong Wu [a], Mengxia Liang [b], Xiaolong Wang [a,*], Qingcai Chen [a]

[a] College of Computer Science and Technology, Harbin Institute of Technology, Shenzhen 518055, China
[b] Faculty of Computing, Harbin Institute of Technology, Harbin 150001, China

ARTICLE INFO

Keywords:
Time series classification
Non-Euclidean structure data
Visibility graph
Ensemble learning
Graph-based features

ABSTRACT

Time series classification is an essential part of time series analysis research and has attracted generous researchers' attention. Representation learning and feature space expansion are vital issues in time series classification. A good representation can reveal the hidden information of the time series, while features could improve the performance of the classifier. Most proposed algorithms are based on the Euclidean structural representation of the time series. Inspired by the complex network analysis, it is realized that important structural features can be highlighted and characterized by non-Euclidean structural representation. Due to the lack of extensive evaluation on applying non-Euclidean structural representation and complex network theory to time series classification, an ensemble learning framework based on visibility graph representation (VGbel) is proposed to provide further exploration. Firstly, a directed series-to-graph transformer is constructed based on the visibility graph algorithm to map the time series into a non-Euclidean space. Secondly, the directed graph representation is subjected to multiscale feature extraction and is fused with the statistical features of the original time series. Furthermore, a hybrid enhanced ensemble model based on stacking is proposed for more stable performance. The proposed method has been extensively evaluated on UCR time series archive, and the experimental results demonstrate good classification accuracy and competitiveness.

1. Introduction

With the development of new technologies such as the web, the Internet of Things, and data collection systems, the quantity of data generated has increased significantly, a majority of which is in the form of time series (Mahdavinejad et al., 2018). Time series (Wei, 2013) is a model based on the information provided by a finite number of observations, also known as a sequence taken at continuous equally spaced points in time order. Time series classification (TSC) is considered as one of the most challenging tasks in data mining and time series analysis. Time series classification is to classify and organize the time series for better analysis and decision-making, whose target is to generate a mapping from the time series to the category labels (Bagnall, Lines, Bostrom, Large, & Keogh, 2016; Fawaz, Forestier, Weber, Idoumghar, & Muller, 2019; Geurts, 2001). Most time series classifications are supervised tasks, as expert knowledge is usually used to construct the category labels (Dau et al., 2018).

In the past decade, many researchers have proposed hundreds of algorithms to solve the time series classification problem from various perspectives. Generally, these algorithms can be divided into two categories: whole-series-based (also called distance-based) and feature-based (Bai, Li, Wang, Wu, & Yan, 2021).

In whole-series-based methods, distance measures over the entire time series are considered, and typically the nearest neighbor classifier (KNN) is used for classification (Lines & Bagnall, 2014). As shown in Fig. 1, there are two types of distance metrics. One is to calculate the Dynamic Time Warping (DTW) distance between each data point and then accumulate it as the distance between two time series; the other is to regard the time series as a high-dimensional vector and map the time series as a point in a high-dimensional space, then calculate the Euclidean Distance (ED) between points (Datta, Karmakar, & Palaniswami, 2020; Xi, Keogh, Shelton, Wei, & Ratanamahatana, 2006). In feature-based methods, raw time series is converted into feature vectors through feature extraction (Lubba et al., 2019) and representation learning, and classification is implemented in the space of feature vectors (Fig. 1).

Even though many algorithms have been proposed to solve problems in time series classification, these methods rely on the Euclidean structural representation, which is one-sided for extracting hidden

* Corresponding author.
E-mail addresses: wushaocong2013@gmail.com (S. Wu), liangmengxia@hotmail.com (M. Liang), xlwangsz@hit.edu.cn (X. Wang), qingcai.chen@hit.edu.cn (Q. Chen).

https://doi.org/10.1016/j.eswa.2023.119942
Received 5 August 2022; Received in revised form 9 February 2023; Accepted 19 March 2023
Available online 23 March 2023

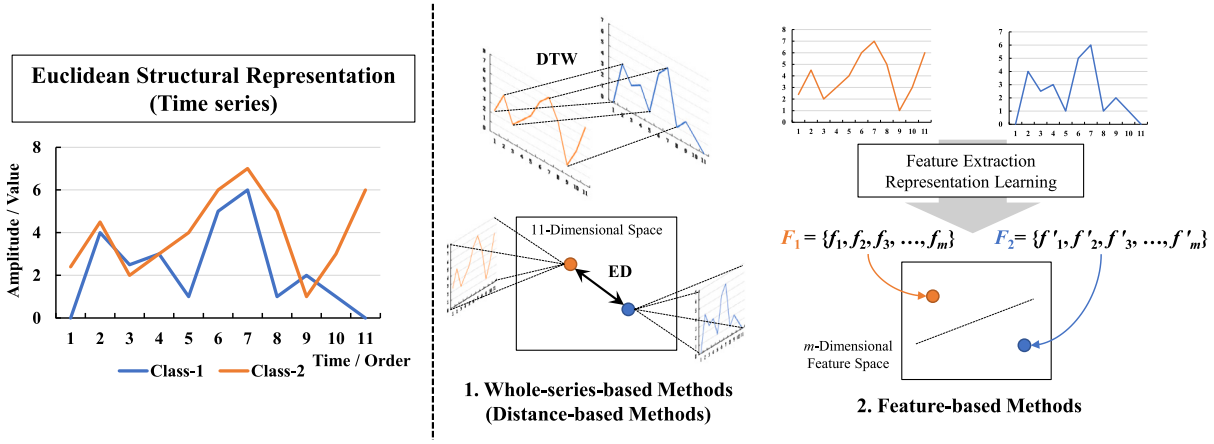


Fig. 1. An illustration of the two types of time series classification methods.

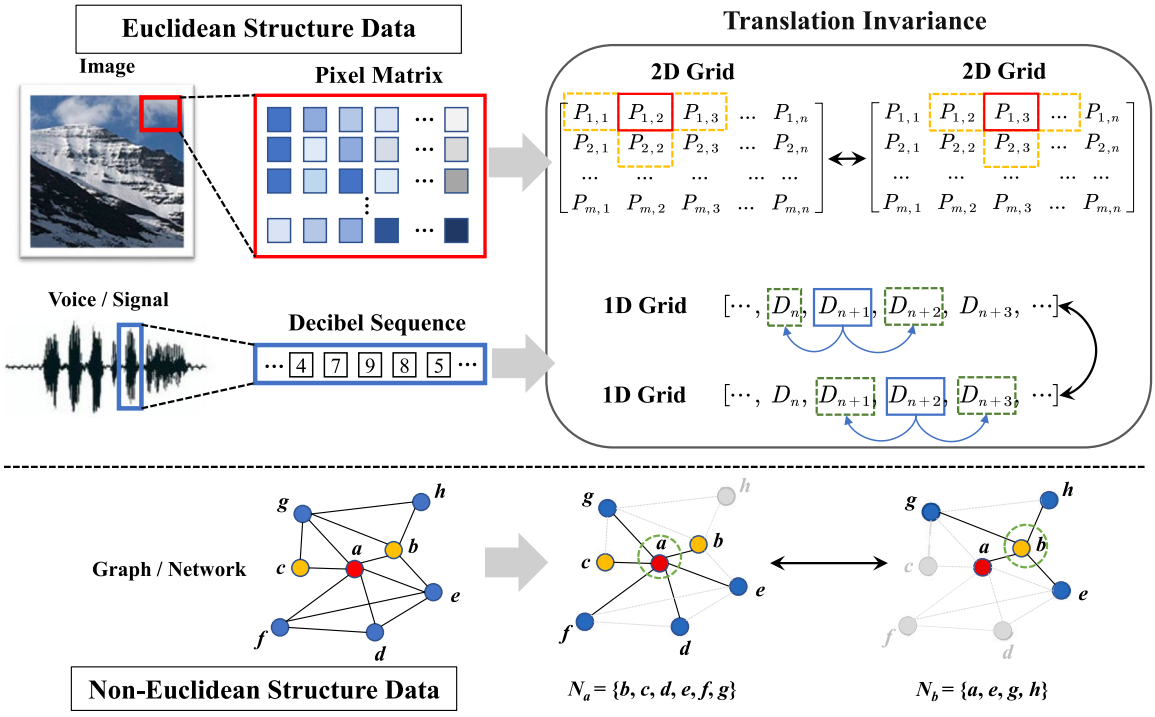


Fig. 2. The difference between Euclidean structured data and non-Euclidean structured data. N_a and N_b denote the neighboring nodes of node a and b, respectively.

structural information. Euclidean structural data is a kind of data with translation invariance (Laub, Roth, Buhmann, & Mller, 2006), typically contains image, text, video, and time series. As shown in Fig. 2, in Euclidean structure data, when one of the data points is centered, the number of its neighbors is always the same, and their positions are relatively fixed. A fixed dimension is one of the characteristics of Euclidean structure data. Similarly, "distance" can be conveniently defined between any two data points in the Euclidean structure (Lele & Richtsmeier, 1991). In other words, in the Euclidean structural representation, the importance of each data point is the same, which is an equalized representation, and the ith data point and the $(i+1)$th data point are highly similar in structure. The association between data points is ambiguous, making it challenging to identify how information interacts between any two data points and how it affects others.

A differentiated representation is necessary to construct correlations between data points, such as a graph, which is classical non-Euclidean structural data. By proposing the visibility graph algorithm (VG), Lacasa, Luque, Ballesteros, Luque, and Nuño (2008) provided a new insight: the structure of a time series is conserved when transforming it into a topological graph, and the graph-based representation is more intuitive in revealing the structural features of the time series. By mapping a time series to a network, structural patterns at different time scales can be visualized from microscopic to macroscale levels.

The visibility graph is considered as the bridge between time series and complex network/graph, in which the structure of the information held by each data point is considered different. Some data points have extremely high weights due to hidden correlations and information interactions, and their distribution provides insight into a particular class's specific structure or pattern. The feature space of a time series can be extended with a graph representation. Moreover, graph representations allow the definition of correlations (or differences) between data points to obtain differentiated information, improving the accuracy and robustness of classification.

The main research questions addressed are summarized below:

- How to obtain the non-Euclidean representation of a time series? How to present the correlations between data points in a non-Euclidean representation?
- How to obtain effective information from the non-Euclidean representation of a time series to distinguish samples in different classes?
- How do ensemble learning methods perform on time series classification tasks? How to obtain a more robust classification accuracy?

By answering these research questions, the contribution of our work can be divided into the following aspects:

- In this paper, a directed series-to-graph transformation was proposed to convert various types of time series in different domains to visibility graph representations, expands the space for feature extraction, and the visibility relation was used to establish a network of correlations between data points.
- Based on the proposed directed series-to-graph transformation, the complexity of the visibility graph transformed from different structural time series were discussed. The graph-based representation provides the basis for mining the discriminative features of time series in non-Euclidean spaces.
- Multiscale features were extracted based on non-Euclidean representations, which include whole-graph-level, edge-level, and node-level. By incorporating these features, information at all levels of granularity would be captured.
- An approach of feature fusion was proposed, which combines the statistical characteristics of the original time series and the multiscale features extracted from the graph-based representation. It covers the deficiency that the traditional VG algorithm loses the amplitude information during the transformation, leading to difficulties distinguishing time series that partially rely on numerical characteristics for classification.
- In our study, two types of ensemble learning methods (Bagging and Boosting) were evaluated on 112 UCR datasets, and their results indicated that classification accuracy varied between the two methods, demonstrating the feasibility of a heterogeneous ensemble model.
- A enhanced heterogeneous ensemble method combining Random Forests (RF), Extra Trees (ET), Adaptive Boosting (Adaboost), and Gradient Boosting Decision Trees (GBDT) was proposed, and experimental results demonstrate that this method can achieve more stable and better classification results.
- The graph-based fusion features and enhanced ensemble model have been extensively evaluated as a whole on 112 UCR datasets and compared with 14 benchmark models and ten SOTA models published recently. The experimental results show that our work is an effective attempt to solve the time series classification problem, which is competitive and interpretable.

The organization of this paper is shown below. As our work involves multiple aspects of research, in Section 2, visibility graphs, ensemble learning, and various TSC methods are summarized and analyzed, representative results from recent years are reviewed, and a gap analysis is performed. Section 3 presents the definition of time series classification, the construction of the visibility graph, and the basic structure of ensemble learning. The directed series-to-graph transformation, multiscale graph-based feature extraction, and feature fusion are discussed in Section 4 and the framework for an enhanced ensemble learning approach. Section 5 provides a detailed analysis and validation of the proposed method. In addition, Table A.1 is presented as a dictionary of abbreviations in Appendix.

2. Related work and gap analysis

Research related to our work is multifaceted and consists of three parts: (1) the visibility graph and its applications on time series analysis; (2) ensemble learning and its applications; and (3) different time series classifiers.

2.1. Visibility graph and its applications on time series analysis

In 2008, Lacasa et al. proposed a novel approach to mapping time series into complex networks by defining visibility relation. The constructed graph inherits several properties of the series in its structure. Their work has led researchers to realize that the structure of time series is conserved in graph topology and that visibility algorithms provide a natural bridge between complex network theory and time series analysis. Several developments have been based on traditional visibility graphs, such as horizontal visibility graphs (HVGs) (Luque, Lacasa, Ballesteros, & Luque, 2009) and multiscale limited penetrable visibility graphs (LPVGs) (Gao, Cai, Yang, Dang, & Zhang, 2016), which focus mainly on different methods of constructing visibility graphs.

Visibility graph algorithms have been widely applied in many fields. Xu, Zhang, and Deng proposed a novel method for constructing weighted complex networks based on the visibility graph algorithm and evaluated the performance of the weighted networks using link prediction. In the study by Kartha, Reshi, Walke, and Dastan (2022), the morphological characteristics of various grown films were systematically studied for the first time using the visibility graph algorithm, and the nature of the interfacial height of the individual surfaces was analyzed by HVG.

In decision-making, it is important to transform basic probability assignments into probability distributions. Chen, Deng, and Cheong (2021) proposed a weighted network method based on the ordered visibility graph, named OVGWP. As opposed to existing ordered visibility graph probabilities, OVGWP takes into account the belief value and the cardinality of basic probability assignment.

A variety of engineering problems can be solved using visibility graph algorithms. There is no doubt that the total variation on a graph is a powerful vertex domain index for measuring the smoothness of graph signals, but its performance is closely related to the underlying graph. Accordingly, Gao and Yu (2020) have proposed a new method for diagnosing rolling bearing faults based on TV HVG feature extraction and martingale distance classification. During construction, tunnel engineers should be aware of the complexity of shield tunneling parameters as the operation affects the project schedule, cost estimates, and safety risks. Zhou, Ding, Zhou, and Skibniewski (2019) addressed the characteristics of shield tunneling parameters by introducing a visibility graph model implemented in a complex network of shield tunneling in metro construction.

Using the Natural Visibility Graph (NVG) method, Iacobello, Scarsoglio, and Ridolfi (2018) mapped the intrinsic structure of the time series into a complex network by mapping the spatiotemporal characteristics of the velocity field of a turbulent channel flow.

These typical applications of the Visibility Graph algorithm have proven Visibility Graph's utility in time series analysis. However, traditional visibility graph algorithms require large amounts of computing resources to achieve the mapping from time series to complex networks. To reduce the complexity of traditional visibility algorithms, Lan, Mo, Chen, Liu, and Deng (2015) proposed a fast transformation algorithm based on the "divide & conquer" strategy.

2.2. Ensemble learning and its applications

Ensemble methods are considered the state-of-the-art solution for many machine learning challenges. Such methods improve the predictive performance of a single model by training multiple models and combining their predictions. Based on the framework, ensemble

Table 1
An overview of nine related papers.

Cite	Aim (Type)	Model/Framework	Comparison models
Fan et al. (2021)	EEG signals, perform fatigue and distraction detection based on the extracted feature. (Time series forecasting)	RF, LightGBM, XGBoost	SVM, KNN, DT, LSTM
Zhang, Ren, Liu, Gao, and Zhu (2021)	Making accurate time series predictions with tiny samples. (Time series forecasting)	GBDT	ARIMA, LSTM, MLP, SVR
Akyuz, Uysal, Bulbul, and Uysal (2017)	Demand forecasting for replenishment. (Time series forecasting)	SES, Holt–Winters Models, ARIMA	None
Qiu, Zhang, Ren, Suganthan, and Amaratunga (2014)	Short-term load demand prediction in power systems. (Time series forecasting)	DBN	SVR, FNN, ENN
Galicia, Talavera-Llames, Troncoso, Koprinska, and Martínez-Álvarez (2019)	Spanish electricity consumption data for 10 years measured with a 10-minute frequency. (Time series forecasting)	DT, GBT, RF	ANN, PSF, DL
Papadopoulos and Karakatsanis (2015)	Day-ahead electricity load forecasts for the ISO-NE CA area. (Time series forecasting)	GBRT	SARIMAX, RF
Wu and Gao (2018)	Forecasting financial time series. (Time series forecasting)	AdaBoost-based LSTM	ARIMA, MLPNN, SVR, ELM, LSTM
Jin and Dong (2016)	Biomedical time series classification on the Chinese Cardiovascular Disease Database. (Time series classification)	Deep neural network-based ensemble method	Bagging methods, Adaboost
Zhong et al. (2020)	Network traffic anomaly detection. (Time series anomaly detection)	HELAD	IF, GMM, SVM, SAE, RBM, Kitsune

learning methods are classified into Bagging (such as RF and ET), Boosting (Adaboost and GBDT), and Stacking (Rincy & Gupta, 2020). While ensemble learning is essentially a model enhancement approach, the training data is learned by multiple sub-classifiers (meta-estimators) to represent probabilistic features (Sagi & Rokach, 2018).

In the analysis of time series, ensemble methods are widely used. Table 1 lists the essential contents of nine representative articles that have been reviewed. Numerous research studies have demonstrated the integration of various models to improve the performance of corresponding tasks significantly.

2.3. Different proposed methods for time series classification

Time series classification is a fundamental problem in data mining and has a variety of applications. Numerous methods have been proposed to address this problem, which can be divided into six categories based on their cores: dictionary-based, distance-based, interval and spectral-based, shapelet-based, deep learning-based, and hybrid methods.

The dictionary-based methods refer to the idea of natural language processing. Researchers believe a time series is a special sentence consisting of discrete words or phrases. The first issue to be addressed is segmenting and mapping the time series into characters. The three main methods of time series symbolization are Piecewise Aggregate Approximation (PAA) (Keogh, Chakrabarti, Pazzani, & Mehrotra, 2001; Keogh & Pazzani, 2000), Symbolic Aggregate Approximation (SAX) (Sun, Li, Liu, Sun, & Chow, 2014; Zhang, Dong, & Xu, 2020), and Symbolic Fourier Approximation (SFA) (Schäfer & Högqvist, 2012). The Bag-of-SFA Symbols (BOSS) method was proposed based on the bag-of-words model (Schäfer, 2014). This method recorded high-frequency symbol features and was used to distinguish different types of time series samples. Middlehurst, Vickers, and Bagnall (2019) and Large, Bagnall, Malinowski, and Tavenard (2019) further proposed Contract BOSS (cBOSS) and Spatial BOSS (S-BOSS). Word Extraction for Time Series Classification (WEASEL) is another dictionary-based method that uses a supervised symbolic time series representation for discriminatory word generation and a Bag of Patterns (BOP) model for building discriminative feature vectors (Lin, Khade, & Li, 2012; Schäfer & Leser, 2017).

The distance between time series is an essential component of many TSC methods. There is a consensus that a time series can be viewed as a point in a multidimensional space whose dimension is determined by the length of the series. Time series of different types will have different aggregations. At this time, distance is an effective means of distinguishing between individuals. The most commonly used methods are K-Nearest Neighbors (KNN) and Elastic Ensembles (EE) (Lines & Bagnall, 2014). Proximity Forests (PF) were proposed by Lucas et al. (2019) to model a decision tree forest in which distance measures were used to partition data. Since most distance calculations are made one-to-one, it is necessary to use samples of equal lengths. In the case of unequal-length sequences, dynamic time warping (DTW) is a robust calculation method that prevents differences in length and shape between sequences (Datta et al., 2020; Górecki & Łuczak, 2014; Xi et al., 2006). By combining KNN and DTW, both could be taken advantage of simultaneously (Forechi, Souza, Badue, & Oliveira-Santos, 2016; Yu, Yu, Hu, Liu, & Wu, 2011).

In reality, different types of time series may have exactly the same statistical characteristics, such as mean, variance, standard deviation, etc (Ryabko & Mary, 2012). To avoid this problem, interval and spectral-based methods emphasize local rather than overall features. The Time Series Forest (TSF) model proposed by Deng, Runger, Tuv, and Vladimir (2013) is based on converting time series into statistical features of sub-sequences and using the random forest for classification. Further, Cabello, Naghizade, Qi, and Kulik (2020) constructed a Supervised Time Series Forest (STSF), an ensemble of decision trees based upon intervals selected through a supervised procedure. Random Interval Spectral Ensemble (RISE) is a popular variant of time series forest. Unlike time series forest, in RISE a single time series interval is used per tree, and the model is trained using spectral features extracted from the series rather than summary statistics. Since the RISE relies on frequency information extracted from the time series, it can be defined as a frequency-based classifier (Lines, Taylor, & Bagnall, 2016).

Shapelet-based methods are inspired by pattern recognition. Shapelets are defined as "subsequences that are in some sense maximally representative of a class" (Hills, Lines, Baranauskas, Mapp, &

Bagnall, 2013). If we set a binary classification, a shapelet could be a discriminant if it is present in most classes and absent from the other class series. However, any subsequence may be distinguishable, and the subsequence length is arbitrary. Therefore, all samples and their subsequences must be checked through a sliding window, which increases the search space for shapelets. Ji et al. (2018, 2019) proposed a fast shapelets selection algorithm to address this issue.

Since the availability of time series data has increased, hundreds of TSC algorithms have been proposed. Deep neural networks (DNNs) are considered an effective tool in many of these approaches. A comprehensive overview of the application of deep learning to time series classification has been provided by He, Zhang, Ren, and Sun (2016), where ResNet has been proven to be a promising classifier that can solve the degradation problem arising from the network. Fawaz et al. (2020) proposed an ensemble of highly scalable deep convolutional neural networks (InceptionTime) based on the Inception-v4 architecture.

Additionally, many methods are difficult to categorize as a specific type, and these methods are uniformly categorized as "Hybrids" in this paper. The Hierarchical Vote Collective of Transformation-based Ensembles (HIVE-COTE) is a heterogeneous meta ensemble for time series classification. Bagnall, Flynn, Large, Lines, and Middlehurst (2020) believed that the key to TSC is carefully considering the data representation, and the most accurate algorithm design is to ensemble classifiers built on different representations. With HIVE-COTE, multiple types of classifiers are combined: each extracting information about a specific aspect of a time series, whether in the time domain, frequency domain, or summarizing intervals within the time series. Even though HIVE-COTE has outstanding accuracy, the cost of training is prohibitive. Ahmed et al. proposed the Time Series Combination of Heterogeneous and Integrated Embedding Forest (TS-CHIEF), which integrates the most effective embeddings of time series developed in the last decade (Shifaz, Pelletier, Petitjean, & Webb, 2020). In terms of accuracy, TS-CHIEF rivaled HIVE-COTE but required a fraction of the running time. In addition to the recent success of convolutional neural networks in time series classification, Dempster, Petitjean, and Webb (2020) found that simple linear classifiers using random convolutional kernels can achieve state-of-the-art accuracy with a fraction of the computational expense of existing methods. This led to the development of ROCKET, a hybrid time series classification method based on kernels. This is a new direction for TSC, which can reduce computational complexity and improve accuracy. A faster version of ROCKET (MiniRocket) as well as an accurate version of ROCKET (MultiRocket) have been developed based on ROCKET (Dempster, Schmidt, & Webb, 2021; Tan, Dempster, Bergmeir, & Webb, 2022).

2.4. Gap analysis

As a result of reviewing and analyzing the relevant literature, we have reached the following conclusions:

- Although researchers have applied the idea of "ensemble" to various tasks, the performance of traditional ensemble learning methods (Random Forest, Adaboost, etc.) on time series classification tasks has been overlooked. Although their performance can be easily verified, their accuracy has only been systematically analyzed in a limited number of articles, and their experimental results on open data are rarely presented.
- In analyzing various time series (geological, construction, meteorological, financial), the Visibility Graph Algorithm and its variants (HVG, Weighted VG, etc.) have been extensively applied (Chen, Deng, & Cheong, 2021; Kartha et al., 2022). A graph-based or network-based representation is more effective for obtaining the structural characteristics of time series. Unfortunately, the applications of these algorithms to time series classification are currently scarce.
- Different classification methods are evaluated on datasets of differing sizes, and the differences are pretty significant. Several methods have been verified on more than 80 UCR datasets (Bai et al., 2021; Lahreche & Boucheham, 2021), whereas others have been verified on only 30 (Chen, Chen, Yang, & Sun, 2021; Liang & Wang, 2021) or even two datasets (Gweon & Yu, 2021). This makes it difficult to compare the performance of many methods on specific datasets.
- Different classification methods emphasize different emphases. Some methods focus on the classification of long time series, while others concentrate on the classification of short time series (Lahreche & Boucheham, 2021). Similarly, binary classifications and multi-classifications are discussed separately.
- Performance is measured using different metrics. Most methods are concerned with accuracy, but some incorporate rank, error rate, and recall.
- Different models require different amounts of training time. For example, HIVE-COTE takes over eight days to train a time series dataset, while ROCKET only requires two hours to train on 85 UCR datasets (Bagnall et al., 2020; Dempster et al., 2020). Testing some methods is difficult due to the long training period.
- Most research on time series classification only compares similar methods. Although these compared methods are not SOTA (HIVE-COTE, ROCKET, etc.), the method's effectiveness is still demonstrated (Bai et al., 2021; Lahreche & Boucheham, 2021). Only a few studies have been extensively compared with various benchmark models.

3. Preliminaries

This section presents definitions and computational methods for time series classification, visibility graph generation, and ensemble learning. A few examples are provided to make these concepts more understandable.

3.1. Time series classification

Time series classification involves categorizing a sequence based on the features that allow it to differentiate itself from other sequences (Wang et al., 2022). A clearer understanding of this concept can be achieved by defining the underlying concepts, such as time series and category labels.

Definition 1 (*Time Series*)**.** Time series $T = \{t_1, t_2, \ldots, t_n\}$ is a value sequence, while $n \in \mathbb{Z}^+$ is a positive integer and the length of T, and only one real-value t_n at each time instant (Lahreche & Boucheham, 2021). The classic one-dimensional definition treats a time series as a string of data arranged chronologically. One-dimensional data sequences have no morphological structure and cannot be displayed on a two-dimensional plane. Therefore, the traditional one-dimensional definition needs to be upgraded to two dimensions in order to explain the calculation process of the visibility graph. The two-dimensional definition of a time series is shown below, where x_n represents the current data point in the nth position in the entire time series, and y_n corresponds to amplitude (value) (Wu, Wang, Liang, & Wu, 2021).

$$T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\} \quad n \in \mathbb{Z}^+. \tag{1}$$

Definition 2 (*Category Labels*)**.** For each time series T, a unique label L is used to indicate the category to which the series belongs. In practice, the label can be either a number, a character or a string. A time series T with a label can be represented as a two-tuple $T = \{L, V\}$, where L is the category label and V is the data sequence $\{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ $n \in \mathbb{Z}^+$.

Definition 3 (*Time Series Dataset*)**.** Time series dataset $D = \{T_1, T_2, \ldots, T_m\} = \{(L_1, L_2, \ldots, L_m), (V_1, V_2, \ldots, V_m)\}$ is a collection of m time series, where the labels $\{L_1, L_2, \ldots, L_m\}$ correspond to the time series $\{V_1, V_2, \ldots, V_m\}, m \in \mathbb{Z}^+$. In a single-label dataset, there is only one label per time series.

Definition 4 (*Time Series Classification*)**.** Time series classification is the process of labeling each time series T with the category to which it belongs through some algorithms or models. Given an unlabeled time series T_{un} and predefined category labels $\{L_1, L_2, \ldots, L_i\}$, the goal of the classification task is to mine features from the raw data of Tun and use a suitable classifier to find a matching label and assign it to T_{un}.

3.2. Visibility graph

The visibility graph is a simple and fast method for converting a time series into a graph (Lacasa et al., 2008). Time series is typical data with the Euclidean structure, and the visibility line serves as a critical medium for mapping the time series to the non-Euclidean structural representation.

Definition 5 (*Visibility Line*)**.** Visibility line, also known as visibility relation. To establish standard visibility criteria more formally, in a time series $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}\ n \in \mathbb{Z}^+$, if any point $c(x_c, y_c)$ between two points $a(x_a, y_a)$ and $b(x_b, y_b)$ could satisfy the Eq. (2), the point a and the point b have the visibility relation, and they could be two nodes connected in the visibility graph.

$$y_c < y_b + (y_a - y_b)\frac{(x_b - x_c)}{(x_b - x_a)}. \tag{2}$$

Fig. 3 illustrates how to determine whether a visibility line exists between two points in a time series of length 4. A histogram is used rather than a line graph representing the time series T to illustrate the visibility line visually. As shown in Fig. 3(a), the visibility line between a and b can be judged to exist if points a and b can see one another, and middle points c_1 and c_2 do not block the line of sight (satisfying Eq. (2)). As shown in the right subfigure, the value of c_2 has been adjusted so that a and b cannot "see" each other, but a can still see c_1 and c_2. Based on the visibility line, the definition of the visibility graph has been improved as follows.

Definition 6 (*Visibility Graph*)**.** For any time series $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}\ n \in \mathbb{Z}^+$, there is a corresponding visibility graph $G = \{N, E\}$, in which N is a set of nodes, each point in T is a node of G, and E is a set of edges. For every two points a and b, if they satisfy the visibility line, nodes a and b could be connected, and then $edge(a, b)$ is added to E. All pairs of points in T are checked, and the pairs satisfied Eq. (2) are connected.

Further discussing the construction of visibility graphs based on the sample shown in Fig. 3, in Fig. 4, G and G' are constructed on the same time series T and T', as T' adjusts the value of point c_2, which makes G' slightly different from G. The time complexity of the basic visibility graph algorithm is $O(n^2)$. In order to transform a time series of size n, we should check all the $n(n-1)/2$ pairs of nodes whether each pair of two nodes can "see" each other (Lan et al., 2015).

3.3. Ensemble learning

For supervised learning algorithms of machine learning, a stable and well-performing model is the goal, but in practice, only models with multiple preferences can be obtained. Ensemble methods are considered high-level applications, while decision trees (DT) are considered basic/weak estimators (Fenton & Neil, 2018). Ensemble methods aim to improve generalization or robustness by combining the predictions of multiple basic estimators. These methods are classified into three types generally: Bagging, Boosting, and Stacking. The Appendix and Fig. A.1 provide detailed descriptions of each type.

4. VG-based ensemble learning framework

In this section, we describe the proposed VG-based ensemble learning framework in detail, composed mainly of three components. The first is the directed series-to-graph transformation, which is the basis of the entire framework and is used to transform time series into non-Euclidean structural graphs. Secondly, there is the multiscale graph-based features extraction and fusion strategy, which describes extracting features from a graph-based representation and integrating them into a fusion feature. The next is an enhanced Stacking classification method whose structure combines the advantages of three ensemble learning methods: Bagging, Boosting, and Stacking. All algorithms are formally described and defined, and detailed examples are given for easy understanding.

Algorithm 1. Directed series-to-graph transform algorithm

Input: Time series $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}, n \in \mathbb{Z}+$
Output: Directed visibility graph $DVG = \{N, E\}$

// Each data point (x_i, y_i) in T is a node N_i in the DVG.
For $i = 1$ to n **do**:
  Add (x_i, y_i) in N as N_i
End for
// Positioning three key points N_{left}, N_{right}, and N_{Max}.
Set subsequence[N_{left}, N_{right}] = [N_1, N_n]
Set $N_{left} = N_1 = (x_1, y_1)$
Set $N_{right} = N_n = (x_n, y_n)$
If the length of subsequence[N_{left}, N_{right}] > 2 **do**:
  For each N_i in subsequence[N_{left}, N_{right}] **do**:
    if y_k in $N_k = (x_k, y_k)$ is the maximum of all y_i in $N_i = (x_i, y_i)$ **do**:
      $N_{Max} = N_k = (x_k, y_k)$
    End if
  End for
End if
// Check the subsequence[N_{left}, N_{right}] and add all directed edges $E(i, k)$ or $E(k, i)$
For each N_i in subsequence[N_{left}, N_{right}] **do**:
  If $N_i \neq N_k$ and N_i can "see" N_k **do**:
    If $i < k$ **do**:
      Add $E(i, k)$ into E
    End if
    If $i > k$ **do**:
      Add $E(k, i)$ into E
    End if
  End If
End for
// Create new subsequences and then iterate them.
Set new subsequence[N_{left}, N_{Max-1}] and subsequence[N_{Max+1}, N_{right}]
Goto Line 5 and **Repeat** Line 5-24 until the length of subsequence $\leqslant 2$
If the length of subsequence[N_{left}, N_{right}] = 2 **do**:
  Add $E(left, right)$ into E
End if
Return $DVG = \{N, E\}$

4.1. Directed series-to-graph transformation

Although the VG algorithm has created a bridge between time series and graphs, there are still some problems that cannot be ignored in practical applications, including computational complexity and the time-ordered property. There is a variation in the length of the time series. The shortest time series in the UCR time series archives contains only 15 data points (from "SmoothSubspace"), whereas, in more than 20 datasets, the time series exceeds 100 or even 200 points. It becomes exponentially more difficult to calculate whether any two data points in a series have a "visibility line" as the time series becomes longer. On the other hand, the graph created by the basic VG algorithm is undirected since a "visibility line" can be either undirected or bidirectional. This flexible relationship will confuse the time-ordered property of each data point.

In this paper, we propose the directed series-to-graph transformation to obtain a directed visibility graph representation *DVG* with

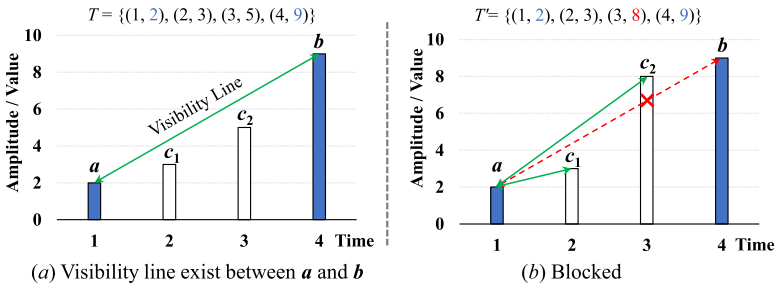


Fig. 3. An example of the visibility line between data points in a time series of length 4.

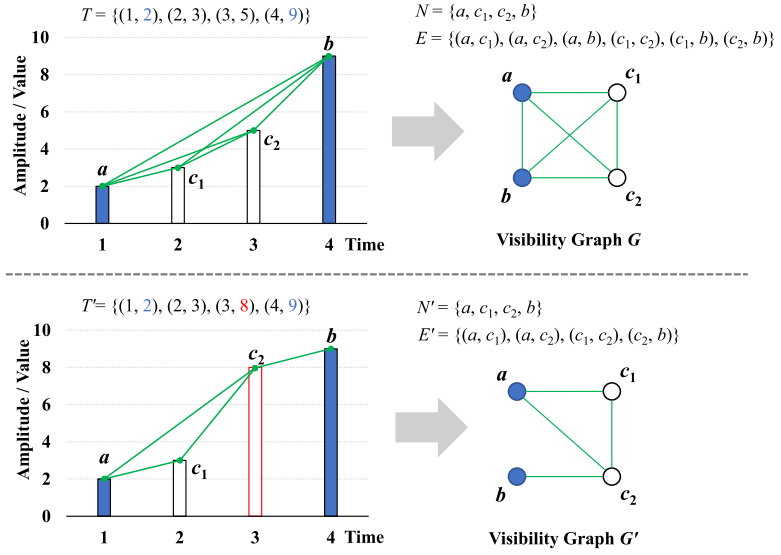


Fig. 4. Schematic illustration of visibility graphs build on different time series.

temporal information. The transformation method is based on the "divide & conquer" strategy proposed by Lan et al. (2015) and limit the time complexity to $O(nlogn)$, retaining only unidirectional edges, which reduces the amount of storage required.

Definition 7 (*Directed Series-To-Graph Transformation*)**.** For any time series $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$ $n \in \mathbb{Z}^+$, there is a corresponding directed visibility graph $DVG = \{N, E\}$, in which N is a set of nodes, each data point (x_n, y_n) in T is an element of N, E is a set of edges. In time series T, x_n is the index of each data point, reflect the order of the corresponding data point. For convenience, assume that $1 \leqslant a < b \leqslant n$, there are any two points in T, $N_a = (x_a, y_a)$ and $N_b = (x_b, y_b)$, "left" and "right" is used to define the location relationship between the two points. N_a is on the left of N_b; N_b is on the right of N_a. N_a, N_b, and the data points between them construct the subsequence $[N_a, N_b]$. As two ends of the subsequence, N_a and N_b can also be called N_{left} and N_{right}. In subsequence $[N_a, N_b]$, if there is a point $N_c = (x_c, y_c)$, $a \leqslant c \leqslant b$, and y_c is greater than y of any data point, then N_c is called the N_{Max}. The process of mapping the T to the DVG is described in Algorithm 1.

Fig. 5 illustrates the directed series-to-graph transformation using a time series T of length 11, where the green dotted line represents the separation between each iteration of the algorithm. In the DVG, 11 nodes and 19 directed edges follow the rule of looking from left to right.

During the transformation from time series to directed graph, the properties of the original time series have been preserved:

- **Connectivity**. There is always a path from the starting point $N_1 = (x_1, y_1)$ to the final point $N_n = (x_n, y_n)$ in DVG (as shown in Fig. 6, the path is highlighted in red). Essentially, each data point can see its nearest neighbor (on the right) under any circumstances and can also be seen by its nearest neighbor on the left.
- **Time-ordered**. In DVG, the temporal characteristics of time series are preserved by constraining the visibility connection, which is reflected in two ways. On one hand, each edge in DVG follows the order from the past to the future. On the other hand, the nodes set N in DVG contains the index/position sequence $[x_1, x_2, \ldots, x_n]$ for all data point. The DVG could be viewed as an expanded representation of a time series.
- **Local characteristics**. As a result of the definition of the visibility line, a few valuable conclusions can be drawn. 2D-Shape of the time series can be divided into two conditions: Peaks and Troughs. It is possible to see the data points on the peaks from the data points on the left or right, but there is no visibility line between the data points on the left or right of the peak due to the structure blocking the sight. Instead, the trough's structure results in a blank sunken area with no barriers, so the visibility lines exist between data points on either side. As shown in Fig. 7, the time series of Brownian motion and its corresponding DVG representation is on the top. The adjacency matrix of the DVG is displayed using the binarized heatmap. Two significant troughs are observed in the Brownian motion time series, which are displayed in the DVG as closely connected cliques (represented by the red and green rectangles).
- **Periodicity and Trend**. As shown in Fig. 7, the classic Box & Jenkins airline data at the bottom contains the monthly totals of international airline passengers from 1949 to 1960. This data shows an increasing trend, non-constant (increasing) variance, and periodic, seasonal patterns. The DVG indicates the periodicity and the trend of the airline series (earlier data points always "see" the later points), which are highlighted in yellow.

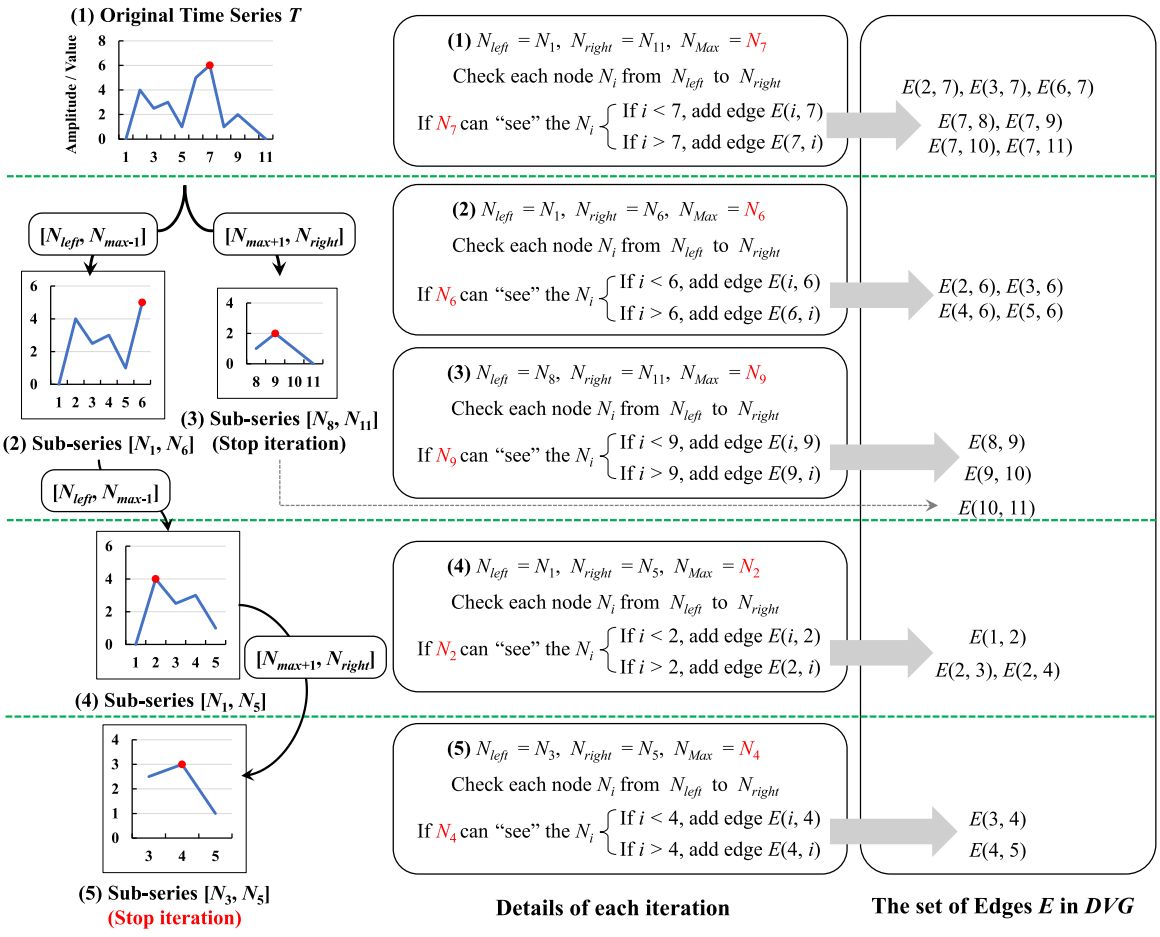


Fig. 5. An example of the directed series-to-graph transformation.

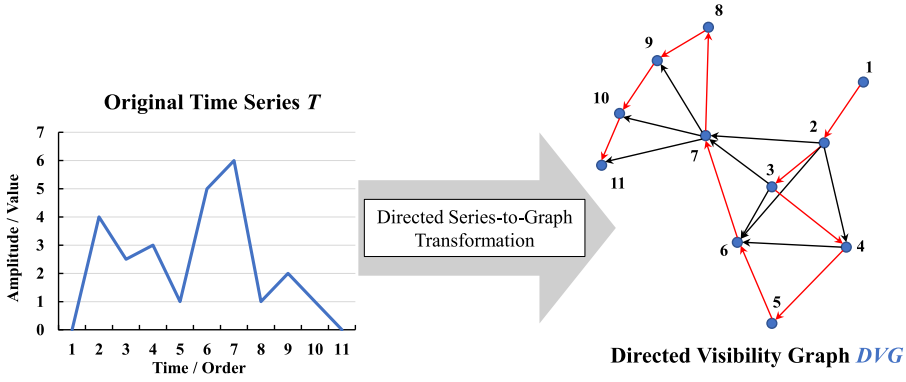


Fig. 6. An illustration of the directed series-to-graph transformation used to convert time series T into *DVG*.

4.2. Multiscale graph-based features extraction and fusion strategy

Complex network theory provides many features that can be used to identify the properties of graphs, such as their centrality and clique distribution. The *DVG* representation satisfies the basic requirements of graph feature extraction. Therefore, we propose a multiscale feature extraction method that extracts features at three levels of the graph: the whole-graph level, the node-level, and the edge-level.

The whole-graph-level feature describes the macroscopic morphologic properties of a graph and represents the overall properties of points and edges. It is feasible to determine the morphology distribution in the time series with the whole-graph-level features. At first, some basic sequences need to be defined to facilitate the subsequent description of the features.

Definition 8 (*Basic Sequences*)**.** For any $DVG = \{N, E\} = \{\{(x_1, y_1), \ldots, (x_n, y_n)\}, \{E_1, \ldots, E_m\}\}$, there are three basic sequences: *ks*, *cs*, and *ps*. *ks* is a sequence that contains the degree of each node in the *DVG*. The distribution of degrees is not continuous, and *ks* only records degrees that have appeared. *cs* is a sequence that $cs[i]$ is the number of nodes in *DVG* that have degree $ks[i]$. *ps* is a sequence that $ps[i]$ is the empirical probability that a node in *DVG* has degree $ks[i]$. Even in the same time series dataset, different *DVG* have different *ks*. The length-varying basic

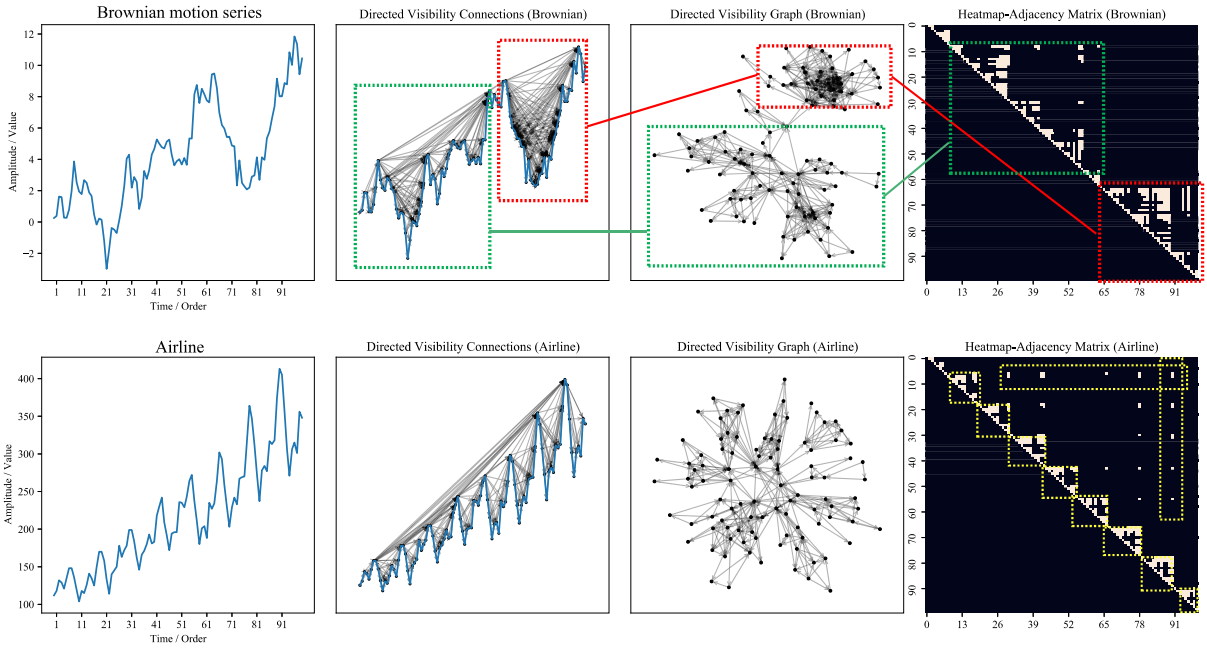


Fig. 7. An illustration of different time series conversion to a directed visibility graph. Heatmaps were used to visualize the adjacency matrix.

sequences cannot be directly used, while their statistical features can prevent this from occurring.

Based on these basic sequences, the following whole-graph-level features were extracted.

- The length of *ks* (*Len_ks*), is a rough statistical characterization of the degree distribution of the nodes in the *DVG*.

$$Len\_ks = Length\,(ks) = k. \tag{3}$$

- The Maximum degree in *ks* (*Max_ks*) and the Minimum degree in *ks* (*Min_ks*) denote the extreme cases of node degree in *DVG*.

$$Max\_ks = Max\,(ks)\,, Min\_ks = Min\,(ks)\,. \tag{4}$$

- The Maximum in *cs* (*Max_cs*), indicates the maximum number of nodes with the same degree in the *DVG*.

$$Max\_cs = Max\,(cs)\,. \tag{5}$$

- The average of *cs* (*Avg_cs*), defines the average number of nodes with the same degree.

$$Avg\_cs = \frac{\sum_{i=1}^{k} cs\,[i]}{k}. \tag{6}$$

- The Maximum probability in *ps* (*Max_ps*), similar to *Max_cs*, denotes the maximum empirical probability of the degree distribution in *DVG*.

$$Max\_ps = Max\,(ps)\,. \tag{7}$$

- The size of the largest clique (*Size_C*) in *DVG*. A clique C in graph is a subset of the nodes, $C \subseteq N$, so every two distinct nodes are adjacent. *Size_C* indicates the number of nodes contained in the largest clique in *DVG*. In addition, finding the largest clique within a graph is an NP-complete problem, so most algorithms have an exponential running time. Although it is difficult to compute this feature, the size of the clique can provide a visual indication of the connectivity of local nodes and reflect fluctuations within the original data.
- The graph transitivity (*DVG_Trans*) is the fraction of all possible triangles present in *DVG*, and the possible triangles are identified by the number of "triads" (two edges with a shared node). Transitivity is the overall probability that the graph has adjacent nodes interconnected, thus revealing the existence of tightly connected communities (clusters, subgroups, cliques). *DVG_Trans* is computed as follows, the number of triangles in a *DVG* is set as Num_T, and Num_t denotes the number of "triads".

$$DVG\_Trans = \frac{3Num_T}{Num_t}. \tag{8}$$

- The average clustering coefficient for *DVG* (*Avg_Clu*) is based on a local clustering coefficient C_i for each node N_i.

$$Avg\_Clu = \frac{1}{n}\sum_{i=1}^{n} C_i = \frac{1}{n}\sum_{i=1}^{n} \frac{Num_T\left[N_i\right]}{Num_t\left[N_i\right]}, \tag{9}$$

where $Num_T[N_i]$ denotes the number of triangles connected to N_i and $Num_t[N_i]$ indicates the number of "triads" around N_i. In graph theory, a clustering coefficient indicates the degree to which nodes are clustered together. This metric emphasizes low-degree nodes, whereas the transitivity ratio emphasizes high-degree nodes. Hence, *DVG_Trans* and *Avg_Clu* are complementary metrics.
- The Wiener index of *DVG* (*WI*). As a metric for describing the structure of molecules, the Wiener index was initially developed in chemistry but was then extended to graphs to quantify diffusivity. According to the formula below, the wiener index of a graph is calculated as the sum of the shortest-path distances between each pair of nodes that can be reached.

$$WI = \frac{1}{n\,(n-1)}\sum_{i=1}^{n}\sum_{j=1}^{n} d_{ij}, \tag{10}$$

where d_{ij} denotes the shortest distance between node N_i and N_j, since *DVG* is a connected graph, this property ensures that any two nodes are reachable and the edges in *DVG* are unweighted, so the shortest distance is equal to the number of directed edges.

As *DVG* is an unweighted graph, each edge only represents the visibility relationship between two nodes, so at the edge-level, we only consider the number of edges (*Num_E*).

$$Num\_E = Length\,(E)\,. \tag{11}$$

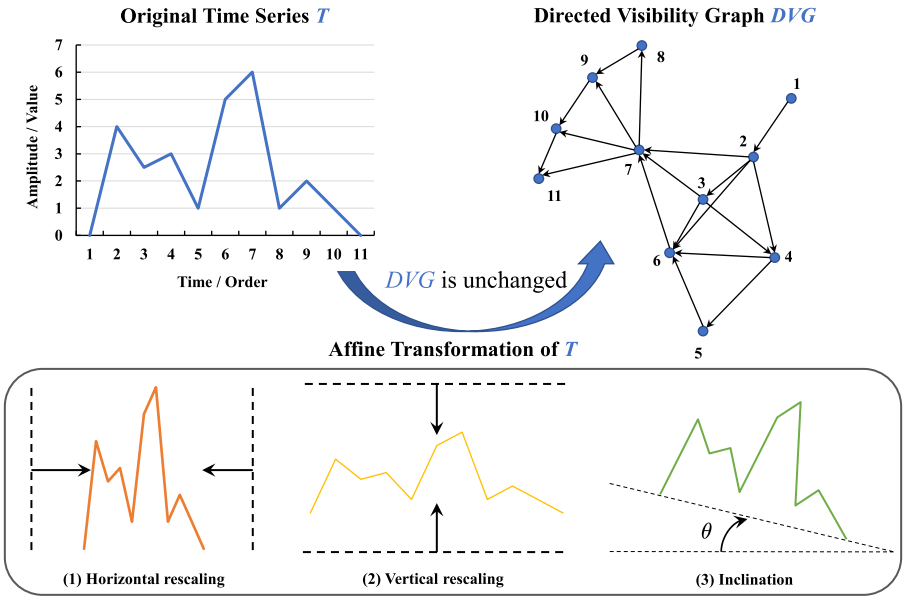


Fig. 8. The directed visibility graph remains invariant under several transformations of the time series.

The weights of each data point are consistent in the time series, which is also a property of the Euclidean structure. Our method for extracting node-level features is based on the differentiation of weights between points based on the *DVG*, and we represent the weights owned by data points using the following five features:

- The degrees of n nodes of *DVG* ($Deg[N_i]$).

$$Deg\left[N_i\right] = Deg_{in}\left[N_i\right] + Deg_{out}\left[N_i\right]. \tag{12}$$

- The ratio of in-degrees to out-degrees ($Ratio^{in}_{out}$) is used to further distinguish between nodes of the same degree. In particular, the in-degree of starting point N_1 is set to 1. Similarly, for the final point N_n, the out-degree is set to 1.

$$Ratio^{in}_{out}\left[N_i\right] = \frac{Deg_{in}\left[N_i\right]}{Deg_{out}\left[N_i\right]} \quad (1 \leqslant i \leqslant n). \tag{13}$$

Centrality is a metric used in Graph Theory and Network Analysis to determine the importance of nodes within a network and quantifies their importance. With the transformed visibility graph representation, it is possible to calculate which data points are more significant than others using a series of centrality measures. There are three types of centrality that we compute in our work.

- Degree centrality (C_D) is the most commonly used metric that assigns weights by the number of edges connected to each node.

$$C_D\left[N_i\right] = \sum_{j=1}^{n} x_{ij} = \frac{Deg\left[N_i\right]}{n-1}, i \neq j. \tag{14}$$

- Closeness centrality (C_C) is computed as follows, where d_{ij} denotes the shortest distance between node N_i and N_j.

$$C_C\left[N_i\right] = \frac{n-1}{\sum_{j=1}^{n} d_{ij}}, i \neq j. \tag{15}$$

- The calculation of Betweenness centrality (C_B) follows Eq. (16), where $\sigma_{jk}\left[N_i\right]$ represents the number of shortest paths passing through N_i between node N_j and node N_k, and σ_{jk} represents the number of all shortest paths from node N_j to node N_k.

$$C_B\left[N_i\right] = \sum_{j \neq i \neq k}^{n} \frac{\sigma_{jk}\left[N_i\right]}{\sigma_{jk}}. \tag{16}$$

Since all the node-level features have the same length, we combine them into a feature matrix of size $5 \times n$ called the Weighted Node Feature Matrix (*WNFM*) as follows.

$$WNFM = \begin{bmatrix} Deg\left[N_1\right], & Deg\left[N_2\right], & \cdots, & Deg\left[N_n\right] \\ Ratio^{in}_{out}\left[N_1\right], & Ratio^{in}_{out}\left[N_2\right], & \cdots, & Ratio^{in}_{out}\left[N_n\right] \\ C_D\left[N_1\right], & C_D\left[N_2\right], & \cdots, & C_D\left[N_n\right] \\ C_C\left[N_1\right], & C_C\left[N_2\right], & \cdots, & C_C\left[N_n\right] \\ C_B\left[N_1\right], & C_B\left[N_2\right], & \cdots, & C_B\left[N_n\right] \end{bmatrix}. \tag{17}$$

With feature fusion, it is possible to obtain the most differentiated information from the multiple original features involved, eliminating redundant information resulting from correlations between features and enabling further decisions. As the network structure of *DVG* is defined exclusively by the binary adjacency matrix of the connections, certain information regarding the actual value of each observation is unavoidably lost. However, this limit applies to all nonparametric rank-based methods, and its invariance under affine (monotonic) transformations makes it insensitive to rescaling. As shown in Fig. 8, the invariance of *DVG* for the three affine transformation cases is visualized. A feature fusion strategy is proposed to bridge this gap by incorporating amplitude and value information from the original time series.

In our strategy, two feature fusion pathways are considered simultaneously. Data points in the original time series are converted into nodes in the *DVG* representation, and their values can be considered inherent weights. Thus, the value sequence $[y_1, y_2, \ldots, y_n]$ of the time series T is added to the *WNFM* to form the Weighted Feature Fusion Matrix (*WFFM*) as Eq. (18).

$$WFFM = \begin{bmatrix} Deg\left[N_1\right], & Deg\left[N_2\right], & \cdots, & Deg\left[N_n\right] \\ Ratio^{in}_{out}\left[N_1\right], & Ratio^{in}_{out}\left[N_2\right], & \cdots, & Ratio^{in}_{out}\left[N_n\right] \\ C_D\left[N_1\right], & C_D\left[N_2\right], & \cdots, & C_D\left[N_n\right] \\ C_C\left[N_1\right], & C_C\left[N_2\right], & \cdots, & C_C\left[N_n\right] \\ C_B\left[N_1\right], & C_B\left[N_2\right], & \cdots, & C_B\left[N_n\right] \\ y_1, & y_2, & \cdots, & y_n \end{bmatrix}. \tag{18}$$

Statistical features have been shown to provide adequate category information, nine statistical features are used in our work:

- Arithmetic mean ($\bar{y}$), This is a rough feature used to describe the average level of all data in the sequence. The calculation of the arithmetic mean follows Eq. (19).

$$\bar{y} = \frac{1}{n}\sum_{i=1}^{n} y_i. \tag{19}$$

- Minimum (y_{min}), maximum (y_{max}) and median value (y_{med}) of a time series T.
- The sum of each data in a time series T. The calculation of this feature is shown below.

$$Sum(T) = \sum_{i=1}^{n} y_i. \tag{20}$$

- Skewness (*Skew*). In probability theory and statistics, skewness refers to the asymmetry of the probability distribution of a real-valued random variable. Skewness can be described as the degree of inclination of a shape to one side or the other. *Skew* is calculated as follows for a time series T:

$$Skew = G_1 = \frac{k_3}{k_2^{3/2}} = \frac{n^2}{(n-1)(n-2)} \cdot \frac{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^3}{\left(\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2\right)^{3/2}}. \tag{21}$$

- Kurtosis (*Kurt*). In probability theory and statistics, kurtosis represents the "tailedness" of a real-valued random variable's probability distribution. In general, the standard kurtosis measure is a scaled version of the fourth moment of a distribution. It is important to note that kurtosis and peakedness are not precisely the same. A higher kurtosis indicates that the data contain large deviations or abnormal points deviating from the mean. However, in most cases, when the amplitude of a time series is high, the corresponding kurtosis is also high. The standard unbiased estimator is used to calculate kurtosis G_2 as follows.

$$Kurt = G_2 = \frac{k_4}{k_2^2} = \frac{\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^4}{\left(\frac{1}{n}\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2\right)^2}. \tag{22}$$

- Standard deviation (*STD*). A standard deviation in statistics refers to the dispersion or variation of a set of values. A low standard deviation indicates values tend to be close to the mean, whereas a high standard deviation indicates values are spread out over a wider range. In order to distinguish between frequently fluctuating series and stable changing series, the standard deviation is calculated as follows:

$$\sigma = \sqrt{\left(\sum_{i=1}^{n}\left(y_i - \bar{y}\right)^2\right)/n}. \tag{23}$$

- Mean absolute deviation (*MAD*). Data scattering is measured by *MAD* as the absolute distance between the sample data and its mean in statistics. *MAD* of a time series T can be calculated as follows:

$$MAD(T) = \frac{1}{n}\sum_{i=1}^{n}\left|y_i - \bar{y}\right|. \tag{24}$$

Algorithm 2. Multiscale graph-based features extraction and fusion algorithm

Input: Time series $T = \{(x_1, y_1), (x_2, y_2), \ldots, (x_n, y_n)\}$, $n \in \mathbb{Z}+$
Direct visibility graph $DVG = \{N, E\} = \{\{(x_1, y_1), \ldots, (x_n, y_n)\}, \{E_1, \ldots, E_m\}\}$ $m,n \in \mathbb{Z}+$
Output: Weighted Feature Fusion Matrix (*WFFM*)
Hybrid Feature Matrix (*HFM*)

// Whole-graph-level features extraction from *DVG*
Set Basic sequences *ks*, *cs*, *ps*
Calculate *Len_ks*, *Max_ks*, *Min_ks*, *Max_cs*, *Avg_cs*, *Max_ps*
Calculate *Size_C*, *DVG_Trans*, *Avg_Clu*, *WI*
// Edge-level feature extraction from *DVG*
Set Num_E = length(E)
// Node-level features extraction from *DVG*
Calculate $Deg[N_i]$
Calculate $Ratio[N_i]$
Calculate $C_D[N_i]$
Calculate $C_C[N_i]$
Calculate $C_B[N_i]$
// Value information from T
Set $Y = [y_1, y_2, \ldots, y_n]$
// Statistical features extraction from T
Calculate $\bar{y}$, y_{min}, y_{max}, y_{med}
Calculate *Sum*(T), *Skew*, *Kurt*, *STD*, *MAD*
// Features fusion
SET $WFFM = \{\}$
SET $HFM = \{\}$
Add $Deg[N_i]$, $Ratio[N_i]$, $C_D[N_i]$, $C_C[N_i]$, $C_B[N_i]$, Y into *WFFM*
Add *Len_ks*, *Max_ks*, *Min_ks*, *Max_cs*, *Avg_cs*, *Max_ps*, *Size_C*, *DVG_Trans*, *Avg_Clu*, *WI* into *HFM*
Add $\bar{y}$, y_{min}, y_{max}, y_{med}, *Sum*(T), *Skew*, *Kurt*, *STD*, *MAD* into *HFM*
Return *WFFM*, *HFM*

In the second pathway of feature fusion, we define the Hybrid Feature Matrix (HFM), which combines widely used statistical features with extracted features at the edge-level and whole-graph-level as Eq. (25). In Algorithm 2, the complete calculation process of multiscale graph-based features extraction and fusion is described.

$$HFM_{5\times4} = \begin{bmatrix} Len\_ks, & Max\_ks, & Min\_ks, & Max\_cs \\ Avg\_cs, & Max\_ps, & Size\_C, & DVG\_Trans \\ Avg\_Clu, & WI, & Num\_E, & \bar{y} \\ y_{\min}, & y_{max}, & y_{med}, & Sum(T) \\ Skew, & Kurt, & STD, & MAD \end{bmatrix}. \tag{25}$$

4.3. Enhanced REAG stacking classification model

In addition to mining discriminative information from features, ensemble learning models are capable of performing powerful classification functions. Section 3.3 briefly introduces the three types of models that are most commonly applied. Traditionally, stacking models use KNN, SVM, RF, lightGBM, etc., as primary classifiers and logistic regression (LR) as a meta-classifier. Many studies have demonstrated that using robust and differentiated classifiers as primary classifiers can improve accuracy while maintaining a reasonable computational cost. In light of the advantages of Bagging and Boosting methods, we proposed an Enhanced model called REAG based on the stacking framework to integrate the advantages of the three types of ensemble learning models.

REAG is a stacking model with a two-layer structure. In the first layer, we selected two Bagging classifiers (RF and ET) and two Boosting classifiers (Adaboost and GBDT) as primary classifiers. Based on the experiments presented in Section 5, we have found a discrepancy in the performance of these four classifiers on the same dataset, which is in line with the requirements of the stacking framework for preliminary estimators. Through the learning of the primary classifier, the *MHF* becomes a meta-feature after the first layer, and the dimensionality of the training and test data is changed. Using a simple linear classifier, such as LR, as a meta-classifier is usually sufficient to ensure reasonable interpretability. However, the contribution of meta-classifiers to performance cannot be ignored. For further improvement of the classification performance and to prevent meta-classifiers from becoming bottlenecks, ET and GBDT are selected as meta-classifiers in the second layer.

Algorithm 3. Enhanced REAG stacking algorithm

Input: Time series dataset $D = \{D_{train}, D_{test}\}$,
Random Forest classifier RF, Extra-Tree classifier ET,
Adaboost classifier Ada, GBDT classifier GBDT
Output: Classification result R_{ET} and R_{GBDT}

// Set parameter K for K-fold cross-validation.
Set $K = 4$
// Divide the training set D_{train}
Set D_{train} = {Train_subset, Validation_subset}, Train_subset : Validation_subset = 3:1
// Create, Train and Test the primary classifiers in the first layer
Create new RF, ET, Ada, GBDT
For $i = 1$ to K:
| **Train** RF, ET, Ada, GBDT on Train_subet
| **Test** RF, ET, Ada, GBDT on Validation_subset
| | **Get** RF_valid_i, ET_valid_i, Ada_valid_i, GBDT_valid_i
| **End test**
| **Test** RF, ET, Ada, GBDT on D_{test}
| | **Get** RF_test_i, ET_test_i, Ada_test_i, GBDT_test_i
| **End test**
End for
// Create new training set and test set
Create new training set $D'_{train} = \{\}$
Create new test set $D'_{test} = \{\}$
Create RF_test, ET_test, Ada_test, GBDT_test = {}
For $i = 1$ to K:
| **Add** RF_valid_i, ET_valid_i, Ada_valid_i, GBDT_valid_i into D'_{train}
End for
For $i = 1$ to K:
| **Add** RF_test_i into RF_test
| **Add** ET_test_i into ET_test
| **Add** Ada_test_i into Ada_test
| **Add** GBDT_test_i into GBDT_test
End for
Set RF_test_avg = Average(RF_test)
Set ET_test_avg = Average(ET_test)
Set Ada_test_avg = Average(Ada_test)
Set GBDT_test_avg = Average(GBDT_test)
Add RF_test_avg, ET_test_avg, Ada_test_avg, GBDT_test_avg into D'_{test}
// Create, Train and Test the meta-classifier in the second layer
Create new ET, GBDT
Train ET, GBDT on D'_{train}
Test ET, GBDT on D'_{test}
| **Get** R_{ET} and R_{GBDT}
End test
Return R_{ET}, R_{GBDT}

In addition, we used stratified k-fold cross-validation in training the first layer classifier to avoid out-of-sample performance degradation due to overfitting. In stratified k-fold cross-validation, k cannot exceed the number of samples in each category, and k is set to 4 according to the distribution of samples in UCR datasets. We demonstrate the calculation process of REAG in Algorithm 3, and the model's overall structure is illustrated in Fig. 9.

REAG implements multi-round representation learning for multiscale fused features. As primary classifiers, Bagging and Boosting models re-represent features through resampling or forward-stagewise, while REAG integrates the outputs of all classifiers in the first layer into a new dataset, changing the features' representation again. Multi-round representation learning allows the original multiscale features to be filtered and recombined, improving correlation. Furthermore, REAG achieves stacked generalization with a reasonable weighting of the four primary classifiers.

4.4. The proposed time series classification framework

The whole work is summarized in Fig. 10, where the framework shown is complete with four components: transformation, features extraction, features fusion, and classification. The details are shown below:

(1) As the result of the directed series-to-graph transformation, all samples in the time series dataset D were transformed into the corresponding *DVG* to form the new representation dataset R_D.
(2) The multiscale features are extracted from each *DVG* in R_D to obtain three levels of graph-based features.
(3) The value sequence Y and statistical features are extracted from the original time series dataset D and incorporated into the multiscale features extracted in Step 2 to form the multiscale hybrid features set (defined as *MHF*).
(4) Classification based on *MHF* using RF, ET, Adaboost, GBDT, REAG-ET, and REAG-GBDT.

5. Experimental evaluation and discussion

This section presents the experimental setup, results, and statistical analysis. The proposed approach required extensive evaluation, producing a high volume of results. There is marginal utility in listing detailed results in massive tables. Instead, the summarized results are presented in short tables or statistical graphs.

5.1. Experimental setup

5.1.1. Datasets and runtime environment

Dau et al. (2019) constructed a UCR time series classification archive for the TSC task, including real-world application scenarios from domains such as medicine, meteorology, computer vision, ECG, sensors, and images, for a total of 128 datasets (check details at https://www.cs.ucr.edu/~eamonn/time_series_data_2018/). Most of the classification algorithms proposed after 2018 are evaluated against the UCR archives. We select 112 datasets in UCR Archive and evaluate our model on these datasets. Table A.2 provides details of the selected datasets, which can be checked in Appendix.

These datasets selected include 38 binary classification datasets and 74 multiclassification datasets with time series samples that range from 15 to 2844 in length and include datasets with more than 50 categories, such as "Fiftywords" and "ShapesAll". These datasets were selected based on two criteria. At first, the team that created the UCR TSC archive provided detailed results for 14 benchmark models but did not cover the entire archive, and only datasets that had been officially validated were considered for comparison (Bagnall, Lines, Bostrom, Large, & Keogh, 2017). Furthermore, due to the differences in the datasets used in recent studies, we have followed their selection whenever possible. It should be noted that all datasets were officially divided into training and test sets, and our experiments were strictly conducted according to the results of the division (check details at http://www.timeseriesclassification.com).

This work is implemented in Python using toolkits such as sktime, sklearn, and networkX. A Ubuntu 16.04 OS workstation with an Intel(R) Core(TM) i7-8700K CPU, and 48 GB of RAM was used for all experiments.

As the work in this paper involves a large number of experiments, we have provided detailed experimental results and standard comparison files on GitHub, and it is available at: https://github.com/shaocongWu/Detailed_Results_of_VGbel

5.1.2. Evaluation metrics

The classification accuracy (ACC) is uniformly adopted as the metric in this paper. Many research results have used classification errors (ERR) to show the performance, which we transformed into accuracy to conduct a comparison. ACC and ERR are calculated using the following formula, where the number of time series correctly classified is defined as n_c, and the total number of time series of test sets is denoted by n_t.

$$\mathrm{ACC} = \frac{n_c}{n_t}, \mathrm{ERR} = 1 - \mathrm{ACC}. \tag{26}$$

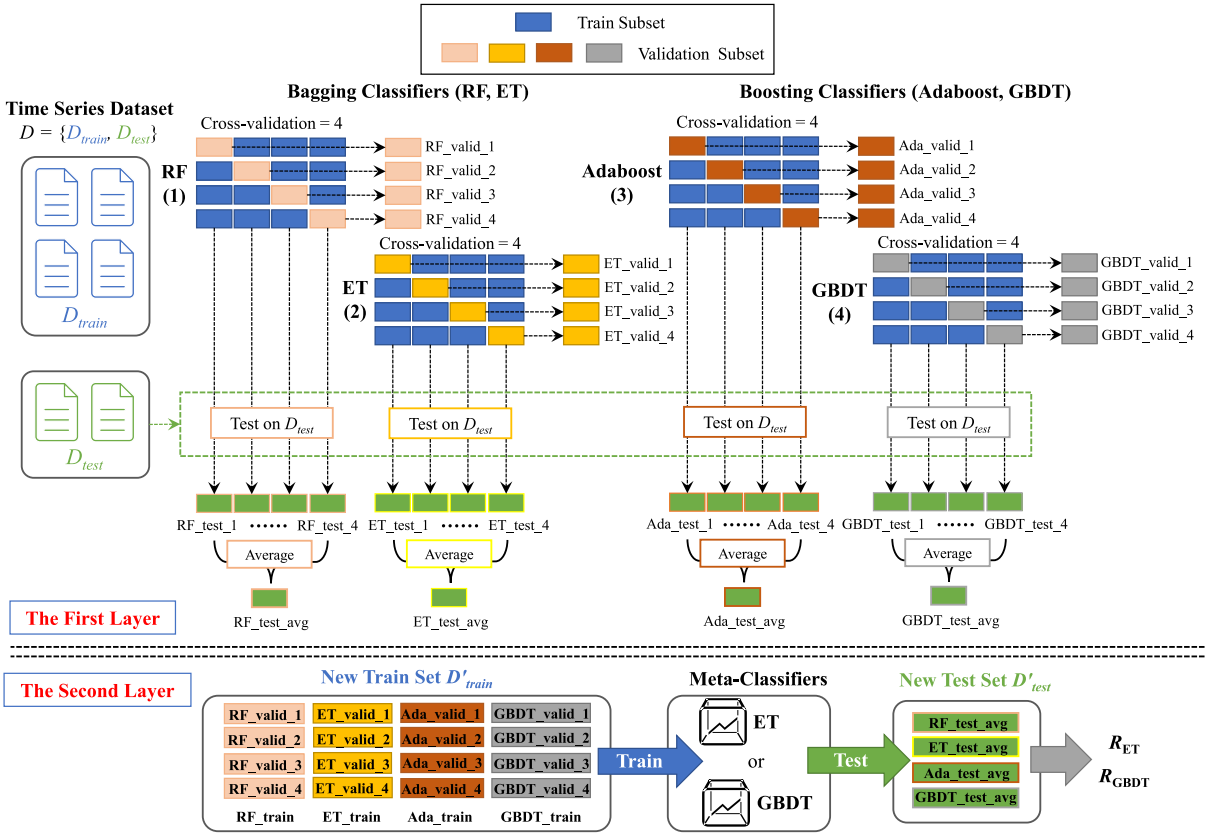


Fig. 9. Schematic diagram of the structure of the REAG model.

Table 2
Detailed parameter settings of the models used in our work. REAG utilizes RF, ET, Adaboost, and GBDT as primary and meta-classifiers, with the same parameters as when these methods are used alone.

	RF	ET	Adaboost	GBDT	REAG-ET	REAG-GBDT
Number of trees	200	200	/	/	200	200
Maximum of estimators	/	/	200	200	200	200
The function to measure the quality of a split	Gini	Gini	/	Friedman MSE	Gini Friedman MSE	Gini Friedman MSE
Boosting algorithm	/	/	SAMME	/	SAMME	SAMME
Learning rate	/	/	0.05	0.05	0.05	0.05
Loss function	/	/	/	Log	Log	Log
k-fold cross-validation	/	/	/	/	4	4

In addition, we measured average accuracy (#Avg. ACC), the number of times the best accuracy was achieved (#Best ACC), and average rank (#Avg. Rank). As our work requires comparing different models on many datasets, we introduce the Critical Difference Diagram (also known as Wilcoxon–Holm post-hoc analysis) for performance analysis (Fawaz et al., 2019). The critical difference diagram is useful for comparing the average rank of multiple models over multiple datasets, and the thick horizontal line identifies models that are not significantly different from each other.

5.1.3. Ensemble learning models and their parameters

For ensemble learning models, careful parameter tuning can significantly improve the performance, but this is not the focus of our work. Therefore, the parameters of the model remain consistent across experiments with arbitrary data sets without targeted adjustments. It is important to note that the models we used were randomized, and each model was run 30 times. Their best results and corresponding states were recorded to ensure reproducible results. The detailed parameter settings are shown in Table 2.

5.1.4. Self-validation scenarios

The purpose of self-validation is to answer two questions:

(1) Compared with the original time series representation, can our proposed non-Euclidean structural representation and the extracted features (*MHF*) significantly improve classification accuracy?
(2) Does the feature fusion strategy make sense? Is there any difference in classification results if only graph-based features are used? If only one of *WFFM* or *HFM* considered?

In order to answer the questions above, two scenarios were set:

(1) Comparing the performance of 6 models using original time series and *MHF* on 112 datasets, respectively. These experiments were conducted to validate the improvement of *MHF* and the enhanced REAG model.
(2) The performance improvement of *MHF* is further discussed by examining four tuned feature inputs: removing all original time series features and keeping only graph-based features, using *WFFM* only, using *HFM* only, and using the complete *MHF*.

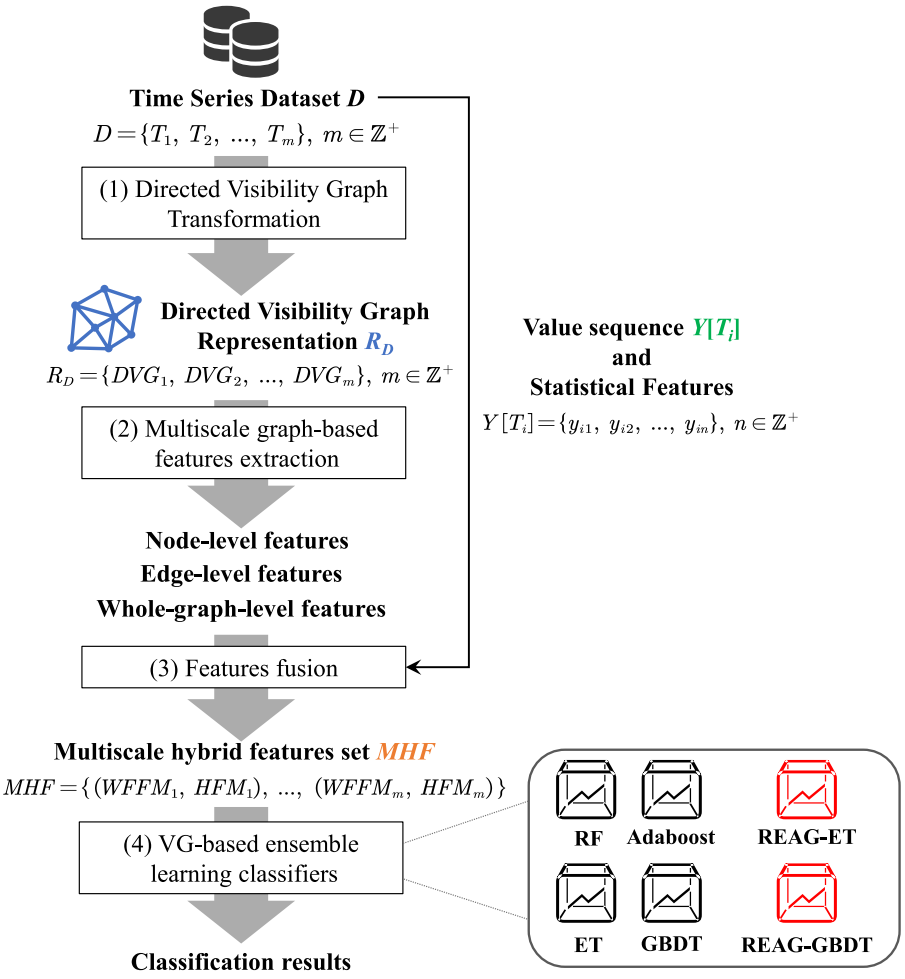


Fig. 10. The complete framework for our work incorporates transformation, feature extraction, features fusion, and classification.

As it would be impractical to adjust the features and re-run the experiments on 112 datasets, we selected ten representative datasets and conducted ablation experiments on them.

5.1.5. Comparative validation scenarios

In this part of the experiment, we take all ensemble learning models (RF, ET, Adaboost, GBDT, and REAG) based on visibility graph representation and features as a whole (VGbel), and their best accuracy is used as a representative for comparison. The comparative verification was divided into two scenarios to evaluate our work comprehensively:

(1) VGbel compared to 14 benchmark models. According to Bagnall et al., the 14 benchmark models included in Table 3 represent the state of the art in 2020. Some of these classifiers, such as ROCKET and HIVE-COTE, are still effective today. By comparing our work with benchmark models, we can quantitatively describe our work's level and performance.
(2) VGbel compared with ten recently published SOTA models. As TSC research has gradually improved, researchers have proposed various models to enhance classification accuracy. Comparing these models can highlight the advantages of VGbel since they are novel and perform well. This paper presents a selection of 10 representative models, whose information is shown in Table 4.

Particularly, the "Multiscale VG" and "CCNF" in Table 4 are among the few studies investigating visibility graphs as a tool for time series classification. Even though "Multiscale VG" was published in 2018, it deserves to be chosen for comparison.

5.2. Self-validation

5.2.1. Macro performance improvement

Since VGbel is a multi-part framework, the experiments in this section will demonstrate the contribution of enhanced REAG models and non-Euclidean representations of features to the improvement of classification performance. We incorporated the concept of control variables with different inputs on the six ensemble learning models (including the two enhanced models, REAG-ET and REAG-GBDT, proposed in this paper). The first is the original time series without any changes (with this input, the model has no prefix), and the second is the proposed multiscale hybrid feature set (*MHF*) based on non-Euclidean representation and feature fusion (in this case, the model will have the prefix "VGbel-" or "VG"). Fig. 11 illustrates the accuracy distribution statistics. The accuracy distribution was divided into six intervals, including 0–0.5, 0.5–0.6, 0.6–0.7, 0.7–0.8, 0.8–0.9, and 0.9–1.0, and the number of data sets was assigned to each interval. The following conclusions can be drawn from preliminary experiments.

- Bagging and Boosting models perform significantly differently regardless of which input is used. Based on the original time series, only a small number of datasets have an accuracy below 0.6 for both RF and ET; the datasets have a concentrated accuracy distribution between 0.7 and 1.0, especially "0.7–0.8" and "0.9–1.0". For Adaboost and GBDT, the number of datasets with an accuracy below 0.6 cannot be ignored (in particular, Adaboost performs below 0.5 on 57 datasets). It was found that when REAG was used, the difference was reduced. However, the

Table 3
Information of 14 benchmark models.

Model Name	Cite	Type	Model Name	Cite	Type
BOSS	Schäfer (2014)	Dictionary Based	Proximity Forest	Lucas et al. (2019)	Distance Based
cBOSS	Middlehurst et al. (2019)	Dictionary Based	ResNet	He et al. (2016)	Deep Learning Based
S-BOSS	Large et al. (2019)	Dictionary Based	InceptionTime	Fawaz et al. (2020)	Deep Learning Based
WEASEL	Schäfer and Leser (2017)	Dictionary Based	HIVE-COTE	Bagnall et al. (2020)	Hybrids
TSF	Deng et al. (2013)	Interval and Spectral Based	TS-CHIEF	Shifaz et al. (2020)	Hybrids
RISE	Lines et al. (2016)	Interval and Spectral Based	ROCKET	Dempster et al. (2020)	Hybrids
STC	Ji et al. (2019)	Shapelet Based	Catch22	Lubba et al. (2019)	Hybrids

Table 4
Information of 10 recently published SOTA models. N_C indicates the number of datasets to be compared.

No.	Model Name	Cite	Type	N_C
(1)	Auto-adaptive multilayer perceptron	del Campo et al. (2021)	Deep Learning Based	61
(2)	TBOPE	Bai et al. (2021)	Dictionary Based	82
(3)	LE-DTW	Lahreche and Boucheham (2021)	Distance Based	28
(4)	MTRL	Chen, Chen, et al. (2021)	Deep Learning Based	36
(5)	GRAE-ESN	Wang et al. (2021)	Deep Learning Based	78
(6)	CSSL	Liang and Wang (2021)	Shapelet Based	25
(7)	MR-PETSC	Feremans, Cule, and Goethals (2022)	Pattern Based	19
(8)	AFFNet	Wang et al. (2022)	Deep Learning Based	85
(9)	Multiscale VG	Li, Lin, Bissyande, Klein, and Le Traon (2018)	Visibility Graph Based	39
(10)	CCNF	Li, Jia, and Wan (2022)	Visibility Graph Based	35

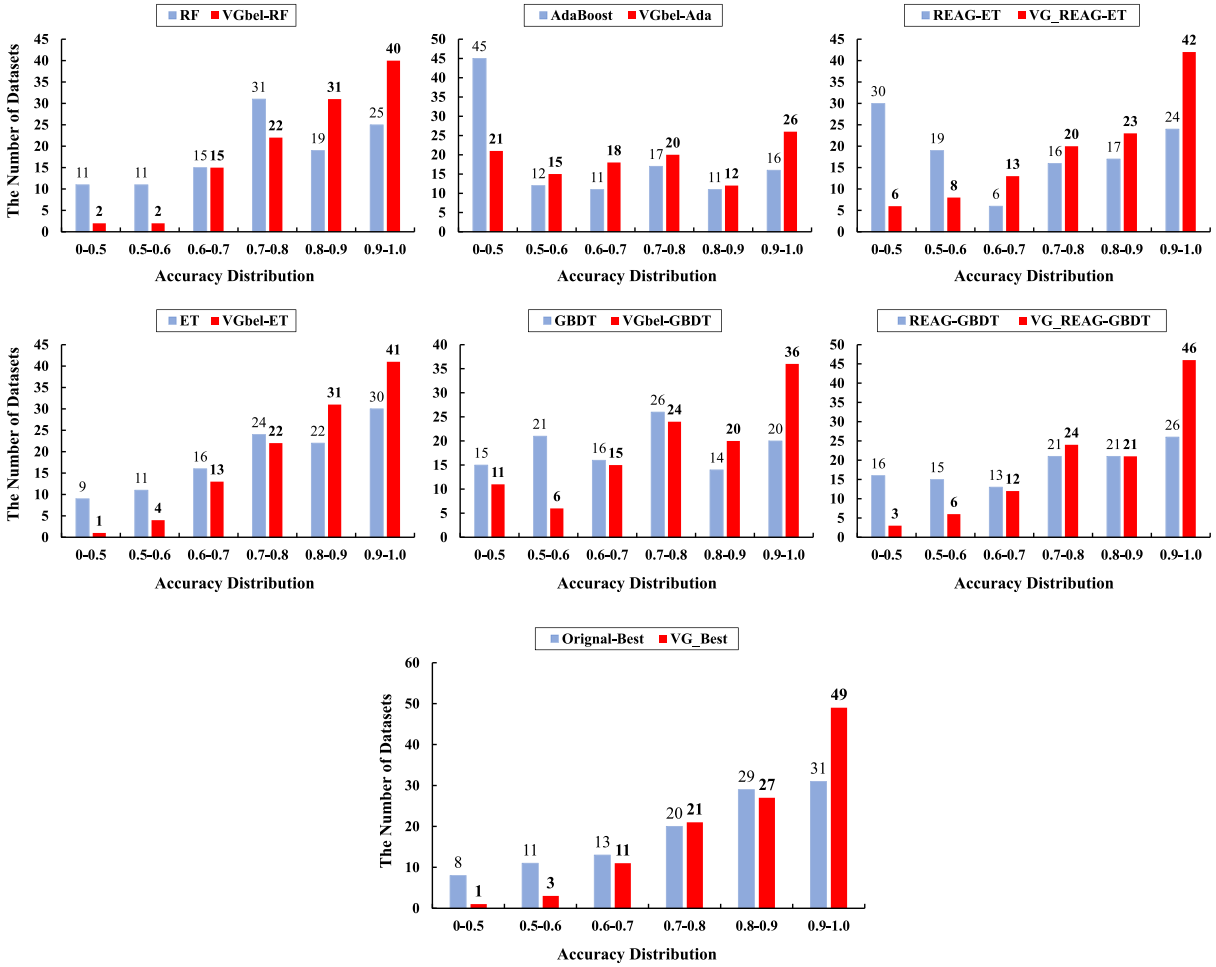


Fig. 11. Results of the accuracy distribution for using raw data and *MHF*. Models using raw time series as input do not have any prefixes (e.g. RF, ET, etc.), while models using *MHF* have the prefix "VGbel" or "VG". In the bottom subplot, the distribution of the best accuracy is shown, which ignores the model used and only considers the different inputs.

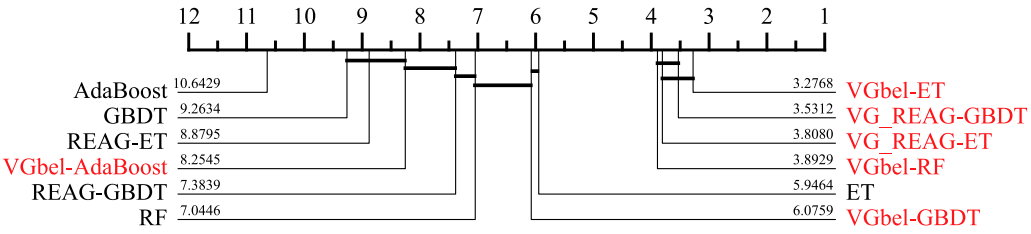


Fig. 12. Critical difference diagram of the comparison with 12 situations using raw time series or *MHF*.

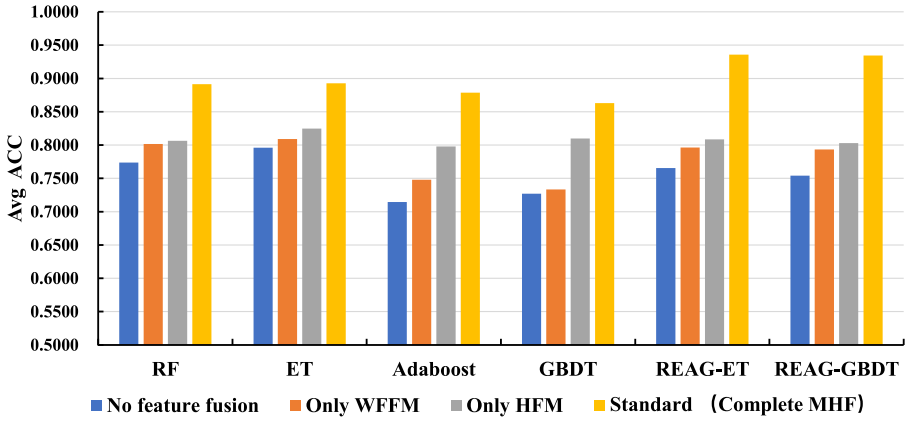


Fig. 13. Average accuracy of 6 ensemble learning models on 10 datasets with different inputs.

different meta-classifiers still led to inconsistent performance for REAG-ET and REAG-GBDT, with the former displaying a greater polarization than the latter.

- There is a remarkable improvement in classification accuracy when *MHF* is used as input. Based on the accuracy distributions of the six models, it is evident that the number of datasets in the low accuracy interval decreases when *MHF* is used as input data, while the number of datasets in the high accuracy interval increases significantly. All models benefit from this enhancement.
- The REAG models used raw time series as input performed mediocrely, even below RF and ET, but its advantages were exploited by learning the information provided by *MHF*. REAG-ET and REAG-GBDT achieved an accuracy of over 0.9 on 42 and 46 datasets, respectively, which is the best performance among all models. However, they perform slightly worse than RF and ET using *MHF* in the 0.7–0.9 accuracy range.
- Based on the accuracy distribution, it is evident that both *MHF* and REAG contribute to improved performance. We used the detailed experimental results to draw the critical difference diagram shown in Fig. 12 to further explore the performance in these situations. Each model is associated with a critical difference; the smaller the critical difference, the better the performance of the input and model. Meanwhile, the models connected by horizontal lines indicate the absence of significant differences between them. Based on Fig. 12, we are confident that the REAG model using *MHF* as input can achieve optimal results in all comparisons. In addition, the four models in the first and the second cliques (VGbel-ET, VG_REAG-GBDT, VG_REAG-ET, and VGbel-RF) are all based on the *MHF* inputs proposed in this paper, which further illustrates the validity of our work.

5.2.2. Discussion on feature fusion

Although *MHF* has been shown to improve classification accuracy significantly, ablation experiments are still necessary to understand how each component of *MHF* impacts the results. We divide *MHF* into four cases for this part of the experiment: no feature fusion (graph-based features only), *WFFM* only, *HFM* only, and complete *MHF*. The Appendix contains detailed experimental results in Tables A.3 and A.4, and Fig. 13 shows the average accuracy of the six models in these four cases. As a result, we can draw the following conclusions.

- In the absence of feature fusion, the performance of the model can be greatly reduced, with accuracy dropping by approximately 15% when only graph-based features are available. This result reinforces our reasoning for proposing feature fusion; the visibility graph is an algorithm with affine invariance that inevitably loses some of the information that is non-negligible and discriminatory during transformation.
- If only *WFFM* or *HFM* were used, performance would improve by about 5%, but this is insufficient to achieve reliable classification. In the Boosting model, a more statistically sophisticated *HFM* achieves more accurate results than a *WFFM*. However, in the Bagging and REAG models, this advantage is minimal.
- It can be observed that the best classification performance is achieved when the complete *MHF* is used, demonstrating that both the non-Euclidean representation and the original time series must be considered.

5.3. Comparative validation

5.3.1. Comparison with benchmark models

VGbel and 14 benchmark models were compared on 108 UCR datasets. According to Table 5, we can conclude that:

- VGbel showed competitive performance in validation experiments involving 108 datasets. VGbel has an average accuracy of 0.8514, higher than ResNet, Proximity Forest, WEASEL, BOSS-like models, RISE, TSF, and Catch22, and only 0.0043 different than STC. Averaging out 15 models, VGbel ranked fifth, behind TS-CHIEF, HIVE-COTE, ROCKET, and InceptionTime. It is noteworthy that

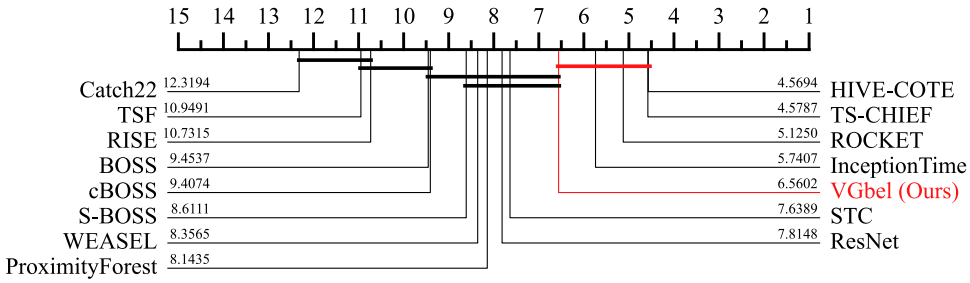


Fig. 14. Critical difference diagram of the comparison with 14 benchmark models on 108 UCR datasets.

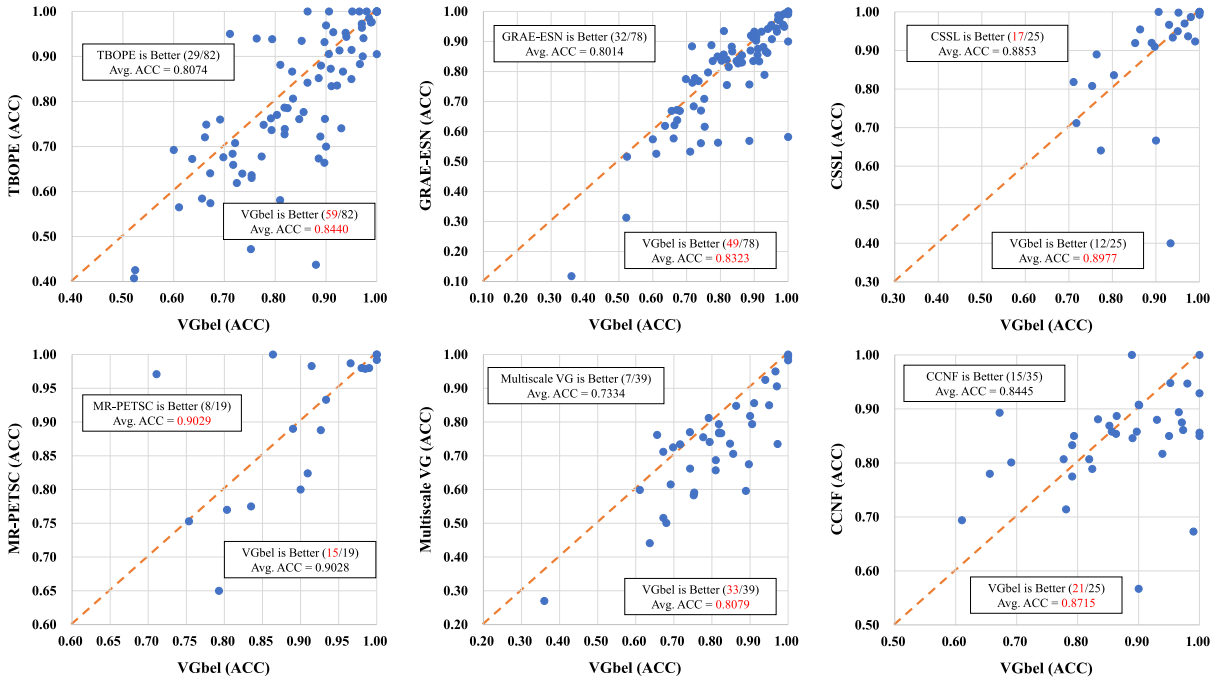


Fig. 15. Pair-wise classification accuracies comparison between VGbel and 6 SOTA models.

VGbel achieved the best results in 33 out of 108 datasets, exceeding the 24 datasets achieved by TS-CHIEF, which ranked second.

- In particular, VGbel achieved the highest accuracy on 16 of the 38 binary classification datasets, exceeding TS-CHIEF, HIVE-COTE (6/38), ROCKET (5/38), InceptionTime (3/38), and ResNet (4/38) by a substantial margin.
- A further distinction can be made between the 14 benchmark models, categorized into traditional models based on specific features (BOSS, Proximity Forest, STC, etc.) and hybrid models (HIVE-COTE, TS-CHIEF, ROCKET). Our model provides more advantages in comparison with traditional methods. Using non-Euclidean representations, VGbel enhances the model's ability to discriminate between time series categories and facilitates time series classification.

Fig. 14 illustrates a critical difference diagram used for further analysis of the results. VGbel was located in the first clique with the best performance after passing the Wilcoxon Signed-Rank test (represented by the red horizontal line), and its performance was not significantly different from the robust classifiers such as ROCKET, TS-CHIEF, HIVE-COTE, and InceptionTime. As a result, we can demonstrate the apparent benefits of using ensemble learning methods for time series classification tasks in combination with non-Euclidean representations.

5.3.2. Comparison with SOTA models proposed recently

In this section, we introduce 10 recently published SOTA models that represent the optimal performance on the TSC task. We have published the detailed experimental results in our GitHub repository, and Table 6 presents the statistical results in the same order as Table 4. Due to the complexity of comparing the models, the following criteria will be followed to ensure a fair comparison.

- Only the experimental results published in the corresponding papers of the model were used in the comparison experiments. It is difficult to reproduce the model and repeat the experiment to ensure the results' accuracy. Accordingly, the experimental results are based on the data disclosed in the paper by the model's developer.
- It may be that some models have been tested on a large number of datasets, but they disclose only the overall statistical results and specific results for only a small number of datasets in their papers. Therefore, only datasets that present detailed results are compared in the paper.
- Sometimes the models to be compared are not optimal, and in their comparison experiments, some models perform equally well or even better, and these models are considered simultaneously to obtain a more comprehensive comparison.

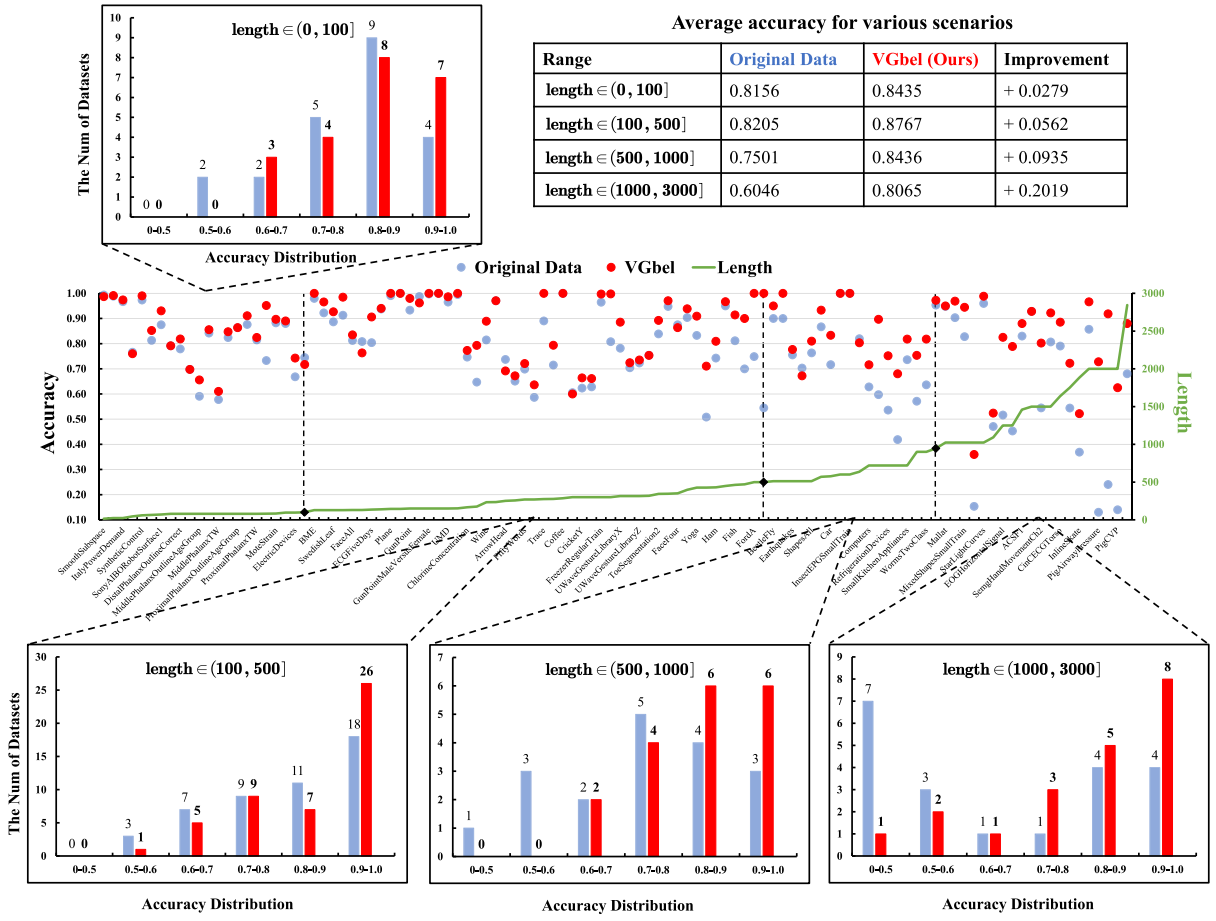


Fig. 16. Accuracy distributions in relation to the length of datasets, and average accuracy over each range.

Table 5
Results of VGbel vs. 14 benchmark models on 108 UCR datasets. The following models are abbreviated: Proximity Forest (PF), HIVE-COTE (H-C), InceptionTime (IT), and TS-CHIEF (T-C). Bold text indicates the best results for each metric.

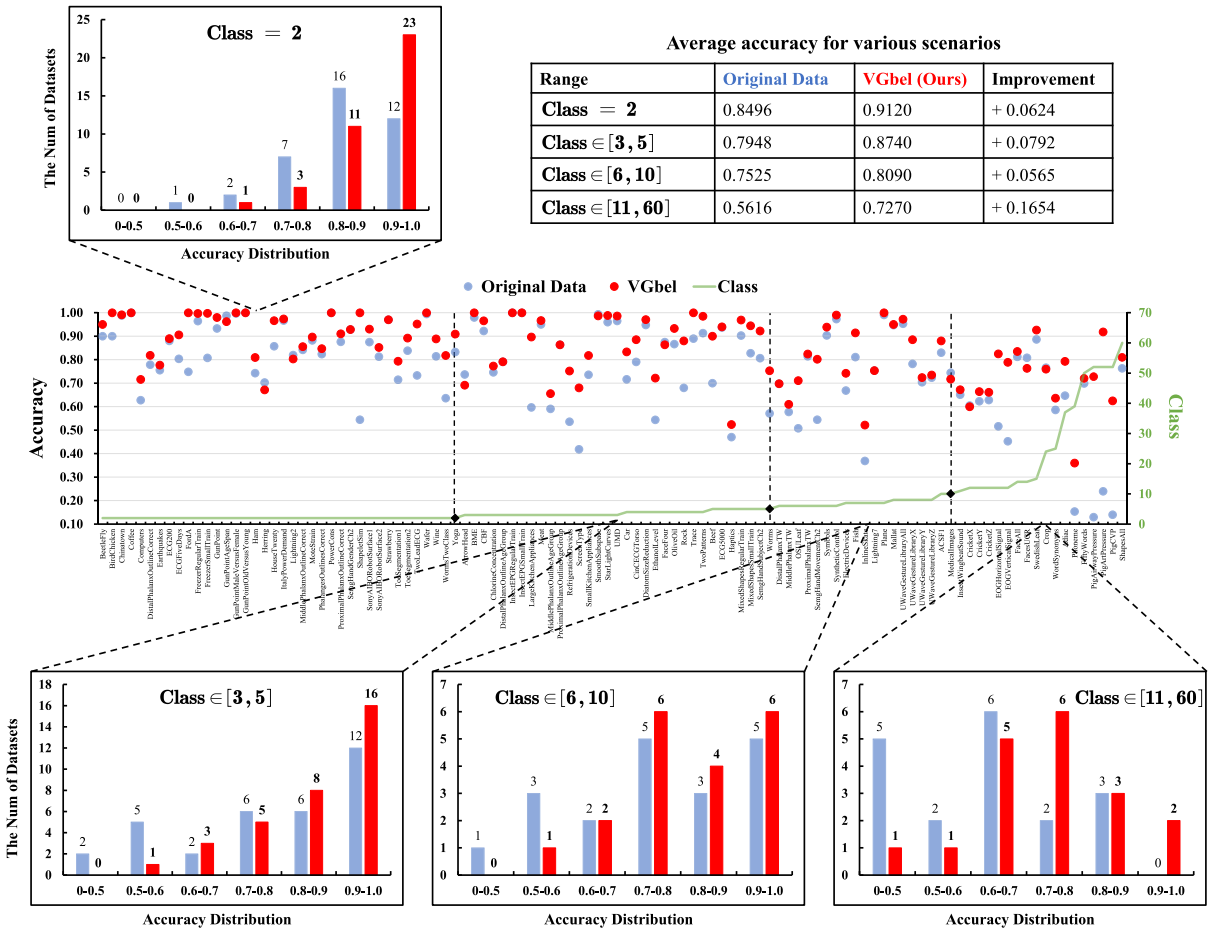
	VGbel (Ours)	STC	ResNet	PF	WEASEL	S-BOSS	cBOSS	BOSS	RISE	TSF	Catch22	T-C	H-C	ROCKET	IT
# Avg. ACC	0.8514	0.8557	0.8292	0.8347	0.8445	0.8426	0.8331	0.8362	0.8006	0.7955	0.7820	0.8757	**0.8782**	0.8640	0.8715
# Best ACC	**33**	5	10	8	3	7	4	2	0	2	0	24	14	16	20
# Avg. Rank	6.38	7.57	7.67	7.96	8.28	8.48	9.24	9.36	10.71	10.86	12.31	4.37	**4.36**	5.02	5.58

- Several models exhibit parameter sensitivity, and the performance varies depending on the parameter. For this reason, results from the same model with different parameters are compared.

Based on the summary and analysis of Table 6, the following conclusions can be drawn.

- VGbel was superior in the majority of comparison experiments, especially in experiments (1), (2), (3), (5), (9), and (10), which totaled six models, showing an overall superiority. There are two deep learning models, Auto-adaptive multilayer perceptron and GRAE-ESN, which demonstrate that features obtained by VGbel orientation are more interpretable than features derived by deep learning.
- According to the experiment numbered (6), VGbel achieved the highest average accuracy, despite not achieving the best score in both the #Best ACC and #Avg Rank metrics. The results indicate that VGbel has a more balanced performance than CSSL.
- In comparison with MTRL, our approach achieved similar performance. Since our method is not parameter-optimized, there is still room for improvement.
- Comparing VGbel with MR-PETSC, VGbel achieved similar average accuracy rates and had a significant advantage in the number of times the best accuracy was obtained and the average ranking.
- Group (8) comparison experiments indicate that AFFNet outperforms VGbel in all metrics due to its better adaptive feature fusion strategy and the use of both multiscale features and distance features. VGbel achieved the best results in 27 out of 85 datasets, better than TSC-FF and EMAN.
- A few models, such as Multiscale VG and CCNF, also employ visibility graphs for time series classification, with their major differences being the methods of feature extraction and whether feature fusion is performed. Both Multiscale VG and CCNF do not account for the information loss caused by the affine invariance of the visibility graph, whereas VGbel specifically incorporates the amplitude information obtained from the original time series, resulting in better classification results.

The pairwise comparison of the accuracy index results for the six one-to-one comparisons is shown in Fig. 15. The orange dashed line was added to make the comparison of the results of the different models more obvious. The performance of VGbel can be observed more directly.



Average accuracy for various scenarios

Range	Original Data	VGbel (Ours)	Improvement
Class = 2	0.8496	0.9120	+ 0.0624
Class ∈ [3, 5]	0.7948	0.8740	+ 0.0792
Class ∈ [6, 10]	0.7525	0.8090	+ 0.0565
Class ∈ [11, 60]	0.5616	0.7270	+ 0.1654

Fig. 17. Accuracy distributions in relation to the class of datasets, and average accuracy over each range.

5.4. Discussion

According to the experiments described in the previous sections, the proposed VGbel is a well-performing and competitive method for time series classification. The following discussion will examine four aspects of VGbel's performance.

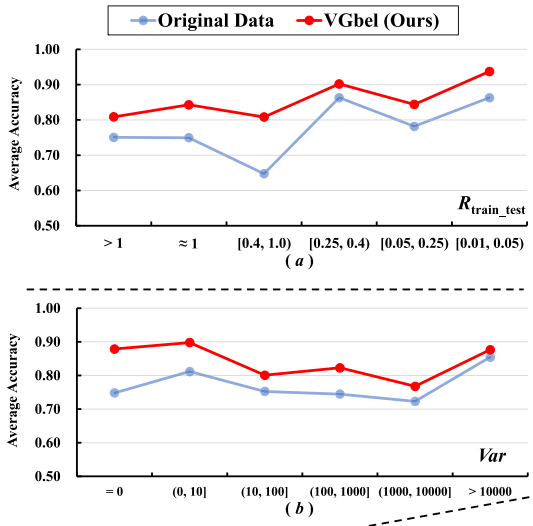
(1) What are the reasons for the disparity between Bagging and Boosting, and how does REAG manage to achieve more stable performance?
(2) What is the relationship between the performance of VGbel and the length of the original time series?
(3) Does the number of categories in the dataset affect the performance of VGbel? Can the binary classification problem be solved more easily than the multicategory problem?
(4) Is the balance of data an influential factor in performance? Does our approach perform differently on balanced and unbalanced datasets?

As for question 1, Bagging and Boosting use completely different optimization methods, where the former resamples the samples, trains a model on each resampled subset, and then averages them. By reducing the effect of outliers and noise, the Bagging model reduces the variance. Boosting is not primarily concerned with variance. Instead, it is concerned with reducing bias through optimal optimization of the loss function. Moreover, since we do not optimize the parameters, the Bagging method can compensate for the limitations imposed by the parameters through the stochasticity. With the stacking framework, REAG integrates two types of algorithms, Bagging and Boosting, to fully use different algorithmic approaches from different data space and structure perspectives, to complement each other's strengths and optimize the final results.

Figs. 16 and 17 illustrate the distribution of accuracy achieved by VGbel for different time series lengths and the number of categories, respectively. In this study, we divided the *length* and *class* of all datasets into four ranges and calculated the average accuracy obtained by VGbel for each range and the improvement compared to the original data.

For the factor of the *length*, there is no significant correlation between it and the performance of VGbel, which obtains an accuracy greater than 0.85 for long time series greater than 1000 while performing poorly (less than 0.7) on some datasets less than 100 in length. It is difficult to assert that VGbel's performance is influenced by the length of the original time series since the length of the time series is independent of its morphological structure information. In addition, our method is robust at different lengths, which means that as the length varies, the average accuracy of the model with original data as input decreases significantly, whereas VGbel remains above 0.8.

The number of categories can determine a task's difficulty. A dichotomous classification task can be judged using the exclusion method, in which if a sample does not belong to one category, it must belong to another, and this method cannot be applied to a multiclassification task. With less than ten categories, VGbel was able to achieve more than 0.85 accuracy, while with more than ten categories, the accuracy reached by VGbel was overwhelmingly below 0.85 or even 0.8. Naturally, any time series classification model must pay close attention to the multiclassification task, and improving the results remains a top priority.



Average accuracy in different unbalanced scenarios

Metric (Range)	Original Data	VGbel (Ours)	Improvement
(*a*) The difference between training and test sets (R_{train_test})			
$R_{train\_test} > 1$	0.7505	0.8086	0.0581
$R_{train\_test} \approx 1$	0.7495	0.8429	0.0934
$R_{train\_test} \in [0.4, 1.0)$	0.6475	0.8083	**0.1608**
$R_{train\_test} \in [0.25, 0.4)$	0.8631	0.9021	0.0390
$R_{train\_test} \in [0.05, 0.25)$	0.7816	0.8440	0.0624
$R_{train\_test} \in [0.01, 0.05)$	0.8630	0.9374	0.0744
(*b*) The difference between the number of samples in each class (*Var*)			
$Var = 0$	0.7482	0.8786	**0.1304**
$Var \in (0, 10]$	0.8117	0.8978	0.0861
$Var \in (10, 100]$	0.7526	0.8007	0.0481
$Var \in (100, 1000]$	0.7447	0.8227	0.0780
$Var \in (1000, 10000]$	0.7230	0.7678	0.0448
$Var > 10000$	0.8544	0.8765	0.0221

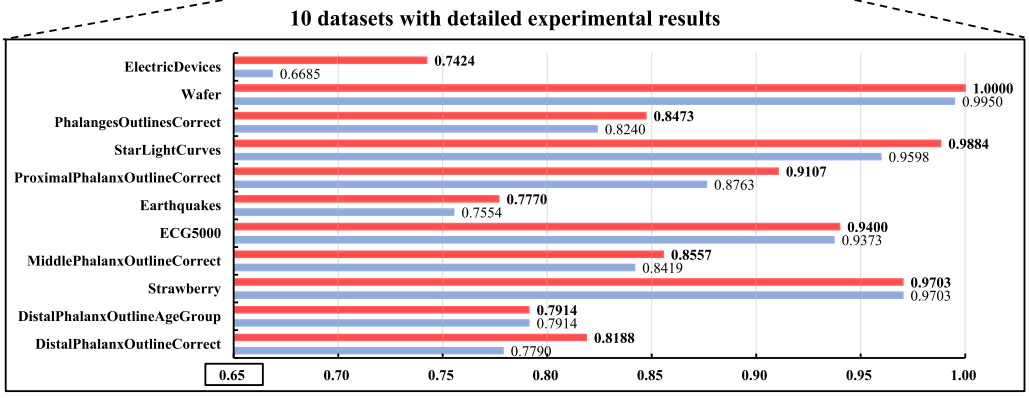


Fig. 18. Discussion of unbalanced datasets. The subplots (a) and (b) represent the two measures of unbalanced datasets in the table. In subplot (c), detailed experimental results are presented for the 10 most unbalanced datasets (*Var* > 10,000).

It is essential to explore the balance of the dataset in detail, and it has become a separate area of research, namely **imbalanced time series classification**. To answer the fourth question, we need to define two variables that describe the degree of imbalance in a dataset. The common criteria for determining whether a dataset is balanced are the "Train-Test" (R_{train_test}), concerned with the ratio between training and test sets, and the "Class-Variance" metric (*Var*), which measures the variation between the number of samples in the various classes of the dataset.

As shown in Fig. 18(a), the average accuracy fluctuates upward as the number of samples in the test set increases or the number of samples in the training set decreases. While this seems counterintuitive, since a decrease in the proportion of training samples implies an increase in the difficulty of training, small sample training can be effective if discriminative features are exploited. According to Fig. 18(b), as the level of 'Class-Variance' increases, the dataset becomes progressively unbalanced, with some classes containing more than ten times of the number of samples than others in severe cases, making it difficult for the model to learn well from these classes during training. A stratified *K*-fold cross-validation training method is employed in our proposed model, which ensures that VGbel can achieve better performance by considering samples from all classes in training and testing. Specifically, Fig. 18(c) shows the detailed results for the ten datasets (*Var* > 10,000), with classification accuracy greater than 0.75 on all datasets except "ElectricDevices".

6. Conclusion and future work

For the TSC task, we present VGbel, a framework based on non-Euclidean representations and ensemble learning models. Based on the visibility graph algorithm, a directed series-to-graph transformation, multiscale feature extraction, feature fusion, and an enhanced stacking model named REAG have been developed. As a result of comparing our work to 14 benchmark models and 10 recently proposed advanced models, we can demonstrate that VGbel is both effective and competitive. As a framework that can be further optimized and tuned, is easy to implement, and uses features and models with good interpretability, VGbel offers a beneficial solution to the TSC problem.

Several shortcomings have been realized: the time series morphology significantly impacts the visibility graph transformation; extracting features from complex graphs is computationally complex. Graph-based feature extraction should be implemented using a faster and more efficient method.

In future work, we intend to use graph neural networks in conjunction to further mine hidden discriminative information with the non-Euclidean representation of the time series. Additionally, XGBoost

Table 6
Results of VGbel vs. 10 SOTA models proposed recently. Bold text indicates the best results.

No.	Model	# Avg ACC	# Best ACC	#Avg Rank
(1)	Auto-adaptive multilayer perceptron (1-layer)	0.7675	2/61	3.36
	Auto-adaptive multilayer perceptron (2-layer)	0.7943	4/61	2.61
	Auto-adaptive multilayer perceptron (3-layer)	0.7937	13/61	2.41
	VGbel (Ours)	**0.8420**	**46/61**	**1.46**
(2)	TBOPE	0.8074	29/82	1.65
	VGbel (Ours)	**0.8440**	**59/82**	**1.28**
(3)	LE-DTW	0.6783	3/28	2.43
	MSM[a]	0.7040	3/28	2.25
	VGbel (Ours)	**0.8172**	**22/28**	**1.25**
(4)	MTRL	**0.8655**	**14/36**	**1.72**
	FCN[b]	0.8316	9/36	2.33
	VGbel (Ours)	0.8621	13/36	1.94
(5)	GRAE-ESN	0.8014	32/78	1.59
	VGbel (Ours)	**0.8323**	**49/78**	**1.37**
(6)	CSSL	0.8853	**17/25**	**1.32**
	VGbel (Ours)	**0.8977**	12/25	1.52
(7)	MR-PETSC	**0.9029**	8/19	1.58
	VGbel (Ours)	0.9028	**15/19**	**1.21**
(8)	TSC-FF[c]	0.8355	20/85	2.54
	EMAN[d]	0.8435	19/85	2.44
	AFFNet	**0.8512**	**39/85**	**2.02**
	VGbel (Ours)	0.8335	27/85	2.58
(9)	Multiscale VG	0.7334	7/39	1.82
	VGbel (Ours)	**0.8079**	**33/39**	**1.15**
(10)	CCNF	0.8445	15/35	1.57
	VGbel (Ours)	**0.8715**	**21/35**	**1.40**

[a]**MSM**: Move–Split–Merge similarity measure (Stefan, Athitsos, & Das, 2013).
[b]**FCN**: Fully Convolutional Network (Wang, Yan, & Oates, 2017).
[c]**TSC-FF**: Time-Series Classification Based on Fusion Features (Wang et al., 2020).
[d]**EMAN**: Echo Memory-Augmented Network (Ma et al., 2021).

and LightGBM are both promising ensemble learning methods that are being considered as potential additions to the framework.

CRediT authorship contribution statement

Shaocong Wu: Conceptualization, Methodology, Software, Validation, Visualization, Formal analysis, Writing – original draft. **Mengxia Liang:** Investigation, Writing – review & editing. **Xiaolong Wang:** Supervision, Project administration, Writing – review & editing, Resources, Funding acquisition. **Qingcai Chen:** Supervision, Resources, Funding acquisition.

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.

Data availability

Data will be made available on request.

Acknowledgments

This work is supported by the Science and Technology Planning Project of Shenzhen Municipality, China [Grant Number JCYJ20190806112210067] and the National Natural Science Foundation of China [Grant Number 61872113]. The authors would like to thank Prof. Eamonn Keogh and all the people who have contributed to the UCR time series classification archive for their selfless work.

Appendix. A brief overview of ensemble learning methods

- **Bagging Method**. Typically, the bagging method considers homogeneous weak estimators, learns these weak estimators independently and in parallel, and then combines them according to some deterministic average process. As a general rule, the combined estimator is more accurate than the single estimator since its variance is reduced. Random forest (RF) (Biau & Scornet, 2016) is a type of bagging method that consists of building a large number of decision trees in order to identify the best set of decision rules. Like the RF model, Extra-Tree (ET) is a perturb-and-combine technique explicitly designed for the tree structure (Geurts, Ernst, & Wehenkel, 2006). Nevertheless, the ET method further enhances its randomness. The features used in RF are randomly selected from the candidate features. However, ET generates thresholds of distinction for each candidate feature and selects the threshold with the highest distinction degree as the segmentation rule. In this way, the variance of the model could be effectively reduced.
- **Boosting Method**. This method is also based on a combination of homogeneous weak estimators. It sequentially learns these weak estimators in a highly adaptive method (each basic estimator attempts to reduce the bias of the combined estimator) and combines them according to a certain deterministic strategy. Several popular boosting methods are available today, including AdaBoost and Gradient Tree Boosting (GBDT). Freund, Schapire, and Abe proposed the former in 1999. In essence, GBDT consists of training a series of weak estimators by repeatedly altering their

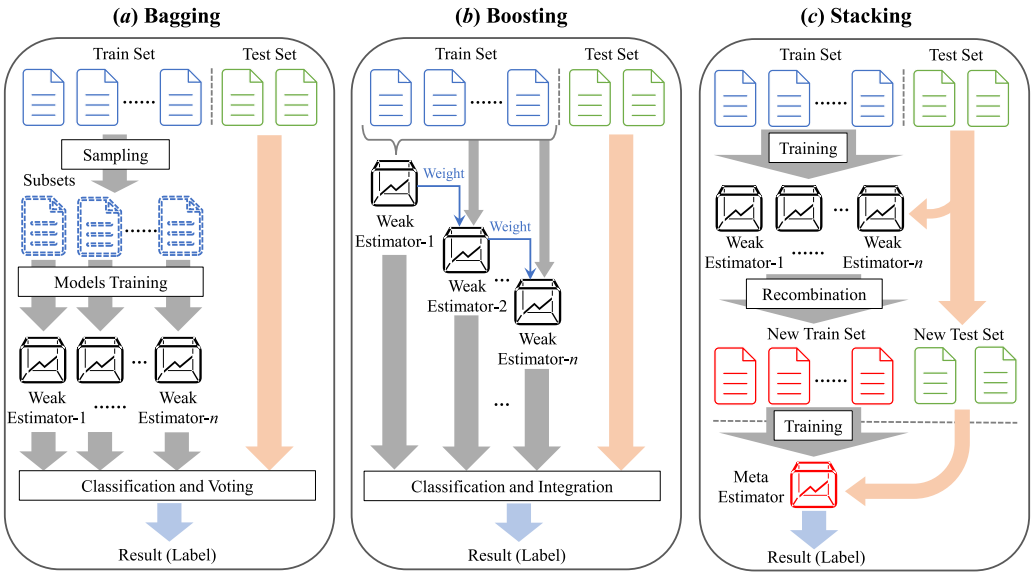


Fig. A.1. Schematic illustration of the framework of three types of ensemble learning methods.

Table A.1
Abbreviations(Abbr) of models/algorithms with corresponding details (In alphabetical order).

Abbr	Detailed Descriptions	Abbr	Detailed Descriptions
AdaBoost	Adaptive Boosting	LSTM	Long Short Term Memory
ANN	Artificial Neural Network	MLP	Multi-Layer Perceptron
ARIMA	AutoRegressive Moving Average	MLPNN	Multilayer Perceptron Neural Networks
BOP	Bag of Patterns	PAA	Piecewise Aggregate Approximation
BOSS	Bag of Symbolic-Fourier Approximation Symbols	PF	Proximity Forest
Catch22	Canonical Time-series Characteristics	PSF	Pattern Sequence-based Forecasting
cBOSS	Contract BOSS	RBM	Restricted Boltzmann Machine fine-tuned neural network
DBN	An ensemble of Deep learning Belief Networks	RF	Random Forests
DNN	Deep learning Neural Network	RISE	Random Interval Spectral Ensemble
DT	Decision Tree	ROCKET	Random Convolutional Kernel Transform
DTW	Dynamic Time Warping	SAE	Sparse AutoEncoder finetuned neural network
EE	Elastic Ensemble	SARIMA	Seasonal AutoRegressive Moving Average
ELM	Extreme Learning Machine	SARIMAX	Seasonal AutoRegressive Moving Average with eXogenous variable
ENN	Ensemble feedforward Neural Network	SAX	Symbolic Aggregate approXimation
ET	Extra Trees	S-BOSS	Spatial BOSS
FNN	Feedforward back-propagation Neural Networks	SES	Simple Exponential Smoothing
GBDT	Gradient Boosting Decision Trees	SFA	Symbolic Fourier Approximation
GBRT	Gradient Boosting Regression Trees	STC	The Shapelet Transform Classifier
GMM	Gaussian Mixture Models	STSF	Supervised Time Series Forest
HELAD	Heterogeneous Ensemble Learning Anomaly Detection model	SVM	Support Vector Machine
HIVE-COTE	The Hierarchical Vote Collective of Transformation-based Ensembles	SVR	Support Vector Regression
HVG	Horizontal Visibility Graph	TS-CHIEF	Time Series Combination of Heterogeneous and Integrated Embeddings Forest
IF	Isolation Forests	TSF	Time Series Forest
KNN	K-Nearest Neighbor	VG	Visibility Graph
LightGBM	Light Gradient Boosting Machine	WEASEL	Word Extraction for Time Series classification
LPVG	Limited Penetrable Visibility Graph	XGBoost	eXtreme Gradient Boosting

weights. Meanwhile, GBDT is a generalization of the lifting algorithm for any loss function that is differentiable. It can be used in many fields for classification and regression, including web search ranking and ecological environment (Friedman, 2002).

Table A.2
Details of the 112 datasets selected in the UCR time series classification archives.

Dataset	Type	Train	Test	Class	Length	Dataset	Type	Train	Test	Class	Length
ACSF1	Device	100	100	10	1460	Mallat	Simulated	55	2345	8	1024
Adiac	Image	390	391	37	176	Meat	Spectro	60	60	3	448
ArrowHead	Image	36	175	3	251	MedicalImages	Image	381	760	10	99
Beef	Spectro	30	30	5	470	MiddlePhalanxOutlineAgeGroup	Image	400	154	3	80
BeetleFly	Image	20	20	2	512	MiddlePhalanxOutlineCorrect	Image	600	291	2	80
BirdChicken	Image	20	20	2	512	MiddlePhalanxTW	Image	399	154	6	80
BME	Simulated	30	150	3	128	MixedShapesRegularTrain	Image	500	2425	5	1024
Car	Sensor	60	60	4	577	MixedShapesSmallTrain	Image	100	2425	5	1024
CBF	Simulated	30	900	3	128	MoteStrain	Sensor	20	1252	2	84
Chinatown	Traffic	20	343	2	24	OliveOil	Spectro	30	30	4	570
ChlorineConcentration	Sensor	467	3840	3	166	OSULeaf	Image	200	242	6	427
CinCECGTorso	Sensor	40	1380	4	1639	PhalangesOutlinesCorrect	Image	1800	858	2	80
Coffee	Spectro	28	28	2	286	Phoneme	Sensor	214	1896	39	1024
Computers	Device	250	250	2	720	PigAirwayPressure	Hemodynamics	104	208	52	2000
CricketX	Motion	390	390	12	300	PigArtPressure	Hemodynamics	104	208	52	2000
CricketY	Motion	390	390	12	300	PigCVP	Hemodynamics	104	208	52	2000
CricketZ	Motion	390	390	12	300	Plane	Sensor	105	105	7	144
Crop	Image	7200	16800	24	46	PowerCons	Power	180	180	2	144
DiatomSizeReduction	Image	16	306	4	345	ProximalPhalanxOutlineAgeGroup	Image	400	205	3	80
DistalPhalanxOutlineAgeGroup	Image	400	139	3	80	ProximalPhalanxOutlineCorrect	Image	600	291	2	80
DistalPhalanxOutlineCorrect	Image	600	276	2	80	ProximalPhalanxTW	Image	400	205	6	80
DistalPhalanxTW	Image	400	139	6	80	RefrigerationDevices	Device	375	375	3	720
Earthquakes	Sensor	322	139	2	512	Rock	Spectrum	20	50	4	2844
ECG200	ECG	100	100	2	96	ScreenType	Device	375	375	3	720
ECG5000	ECG	500	4500	5	140	SemgHandGenderCh2	Spectrum	300	600	2	1500
ECGFiveDays	ECG	23	861	2	136	SemgHandMovementCh2	Spectrum	450	450	6	1500
ElectricDevices	Device	8926	7711	7	96	SemgHandSubjectCh2	Spectrum	450	450	5	1500
EOGHorizontalSignal	EOG	362	362	12	1250	ShapeletSim	Simulated	20	180	2	500
EOGVerticalSignal	EOG	362	362	12	1250	ShapesAll	Image	600	600	60	512
EthanolLevel	Spectro	504	500	4	1751	SmallKitchenAppliances	Device	375	375	3	720
FaceAll	Image	560	1690	14	131	SmoothSubspace	Simulated	150	150	3	15
FaceFour	Image	24	88	4	350	SonyAIBORobotSurface1	Sensor	20	601	2	70
FacesUCR	Image	200	2050	14	131	SonyAIBORobotSurface2	Sensor	27	953	2	65
FiftyWords	Image	450	455	50	270	StarLightCurves	Sensor	1000	8236	3	1024
Fish	Image	175	175	7	463	Strawberry	Spectro	613	370	2	235
FordA	Sensor	3601	1320	2	500	SwedishLeaf	Image	500	625	15	128
FreezerRegularTrain	Sensor	150	2850	2	301	Symbols	Image	25	995	6	398
FreezerSmallTrain	Sensor	28	2850	2	301	SyntheticControl	Simulated	300	300	6	60
GunPoint	Motion	50	150	2	150	ToeSegmentation1	Motion	40	228	2	277
GunPointAgeSpan	Motion	135	316	2	150	ToeSegmentation2	Motion	36	130	2	343
GunPointMaleVersusFemale	Motion	135	316	2	150	Trace	Sensor	100	100	4	275
GunPointOldVersusYoung	Motion	136	315	2	150	TwoLeadECG	ECG	23	1139	2	82
Ham	Spectro	109	105	2	431	TwoPatterns	Simulated	1000	4000	4	128
Haptics	Motion	155	308	5	1092	UMD	Simulated	36	144	3	150
Herring	Image	64	64	2	512	UWaveGestureLibraryAll	Motion	896	3582	8	945
HouseTwenty	Device	40	119	2	2000	UWaveGestureLibraryX	Motion	896	3582	8	315
InlineSkate	Motion	100	550	7	1882	UWaveGestureLibraryY	Motion	896	3582	8	315
InsectEPGRegularTrain	EPG	62	249	3	601	UWaveGestureLibraryZ	Motion	896	3582	8	315
InsectEPGSmallTrain	EPG	17	249	3	601	Wafer	Sensor	1000	6164	2	152
InsectWingbeatSound	Sensor	220	1980	11	256	Wine	Spectro	57	54	2	234
ItalyPowerDemand	Sensor	67	1029	2	24	WordSynonyms	Image	267	638	25	270
LargeKitchenAppliances	Device	375	375	3	720	Worms	Motion	181	77	5	900
Lightning2	Sensor	60	61	2	637	WormsTwoClass	Motion	181	77	2	900
Lightning7	Sensor	70	73	7	319	Yoga	Image	300	3000	2	426
FordB	Sensor	3636	810	2	500	NonInvasiveFetalECGThorax1	ECG	1800	1965	42	750
HandOutlines	Image	1000	370	2	2709	NonInvasiveFetalECGThorax2	ECG	1800	1965	42	750

Table A.3
Detailed results of ablation experiments (No features fusion and Only WFFM).

Dataset	No features fusion						Only WFFM					
	RF	ET	AdaBoost	GBDT	REAG-ET	REAG-GBDT	RF	ET	AdaBoost	GBDT	REAG-ET	REAG-GBDT
BeetleFly	0.8500	0.8000	0.7500	0.4500	0.8000	0.8000	0.8500	0.9000	0.7600	0.4500	0.8700	0.8500
BirdChicken	0.8500	0.9000	1.0000	1.0000	0.9000	0.9500	0.9000	0.8800	0.8700	0.8500	0.9000	0.9000
ECG200	0.7900	0.8100	0.7600	0.8500	0.8200	0.8000	0.8280	0.8640	0.8200	0.8100	0.8360	0.8320
FreezerSmallTrain	0.8060	0.8291	0.7537	0.7305	0.8400	0.8211	0.7465	0.7618	0.8266	0.7945	0.7745	0.7780
GunPoint	0.8667	0.8867	0.7733	0.6800	0.8467	0.8200	0.9507	0.9533	0.7907	0.8240	0.9200	0.9320
ShapeletSim	0.5722	0.6444	0.6444	0.8500	0.6333	0.6111	0.5356	0.5300	0.4889	0.4889	0.4789	0.4978
Yoga	0.7697	0.8003	0.7290	0.7780	0.7787	0.7963	0.7995	0.8268	0.7542	0.8240	0.8289	0.8250
OliveOil	0.8333	0.8667	0.6333	0.6000	0.6667	0.5667	0.9133	0.8800	0.8400	0.8333	0.8667	0.8533
OSULeaf	0.5331	0.5537	0.3182	0.5537	0.5537	0.5496	0.5116	0.5248	0.3512	0.4719	0.5000	0.4826
Chinatown	0.8659	0.8688	0.7843	0.7784	0.8163	0.8251	0.9802	0.9691	0.9778	0.9866	0.9872	0.9825

- **Stacking Method**. The stacking method utilizes heterogeneous estimators, trains them in parallel, and combines them by training a "meta mode" to produce an outcome based on different predictions (Džeroski & Ženko, 2004). Stacking is an ensemble framework based on a hierarchical structure. If we take two-layer stacking as an example, the data set is divided into the training and testing sets first, and multiple primary learners are obtained through training on the training set. Primary learners (weak estimators) are used to predict the testing set, and the output values are used as input values for the next stage of training, which is used for training the secondary learners (meta-estimators). Due to the different training data in the two stages, overfitting can be prevented to a certain degree (see Fig. A.1).

Table A.4
Detailed results of ablation experiments (Only HFM and Standard (Complete *MHF*)).

Dataset	Only HFM						Standard (Complete *MHF*)					
	RF	ET	AdaBoost	GBDT	REAG-ET	REAG-GBDT	RF	ET	AdaBoost	GBDT	REAG-ET	REAG-GBDT
BeetleFly	0.7000	0.7000	0.7500	0.6000	0.7000	0.7000	0.9500	0.9500	0.7500	0.4500	0.9500	0.9500
BirdChicken	0.9500	0.9000	1.0000	1.0000	0.9000	0.9000	0.9500	0.9500	1.0000	1.0000	1.0000	1.0000
ECG200	0.7900	0.8000	0.8200	0.7900	0.7700	0.8100	0.8500	0.8800	0.8300	0.8100	0.8900	0.8900
FreezerSmallTrain	0.7435	0.7660	0.8996	0.8818	0.8123	0.8246	0.8053	0.7919	0.9975	0.9975	0.9975	0.9975
GunPoint	0.8800	0.9133	0.8800	0.8533	0.8533	0.8400	0.9733	0.9733	0.8867	0.9467	0.9800	0.9800
ShapeletSim	0.9500	0.9944	1.0000	1.0000	0.9889	0.9944	0.9556	0.9833	1.0000	1.0000	1.0000	1.0000
Yoga	0.8900	0.8830	0.8980	0.9073	0.8917	0.8973	0.8483	0.8503	0.8873	0.9093	0.9040	0.9040
OliveOil	0.6667	0.7000	0.3667	0.5000	0.6667	0.5333	0.9333	0.9000	0.8667	0.8333	0.9333	0.9333
OSULeaf	0.6777	0.7231	0.4835	0.6859	0.6694	0.7025	0.6653	0.6653	0.5868	0.6942	0.7107	0.6983
Chinatown	0.8163	0.8688	0.8805	0.8805	0.8338	0.8280	0.9825	0.9825	0.9825	0.9883	0.9913	0.9913

References

Akyuz, A. O., Uysal, M., Bulbul, B. A., & Uysal, M. O. (2017). Ensemble approach for time series analysis in demand forecasting: Ensemble learning. In *2017 IEEE international conference on innovations in intelligent systems and applications* (pp. 7–12). IEEE, http://dx.doi.org/10.1109/inista.2017.8001123.

Bagnall, A., Flynn, M., Large, J., Lines, J., & Middlehurst, M. (2020). On the usage and performance of the hierarchical vote collective of transformation-based ensembles version 1.0 (HIVE-COTE v1.0). In *Advanced analytics and learning on temporal data* (pp. 3–18). Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-65742-0_1.

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2016). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*(3), 606–660. http://dx.doi.org/10.1007/s10618-016-0483-9.

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*, 606–660.

Bai, B., Li, G., Wang, S., Wu, Z., & Yan, W. (2021). Time series classification based on multi-feature dictionary representation and ensemble learning. *Expert Systems with Applications*, *169*, Article 114162. http://dx.doi.org/10.1016/j.eswa.2020.114162.

Biau, G., & Scornet, E. (2016). Rejoinder on: A random forest guided tour. *TEST*, *25*(2), 264–268. http://dx.doi.org/10.1007/s11749-016-0488-0.

Cabello, N., Naghizade, E., Qi, J., & Kulik, L. (2020). Fast and accurate time series classification through supervised interval search. In *2020 IEEE international conference on data mining* (pp. 948–953). IEEE, http://dx.doi.org/10.1109/icdm50108.2020.00107.

Chen, L., Chen, D., Yang, F., & Sun, J. (2021). A deep multi-task representation learning method for time series classification and retrieval. *Information Sciences*, *555*, 17–32. http://dx.doi.org/10.1016/j.ins.2020.12.062.

Chen, L., Deng, Y., & Cheong, K. H. (2021). Probability transformation of mass function: A weighted network method based on the ordered visibility graph. *Engineering Applications of Artificial Intelligence*, *105*, Article 104438. http://dx.doi.org/10.1016/j.engappai.2021.104438.

Datta, S., Karmakar, C. K., & Palaniswami, M. (2020). Averaging methods using dynamic time warping for time series classification. In *2020 IEEE symposium series on computational intelligence* (pp. 2794–2798). IEEE, http://dx.doi.org/10.1109/ssci47803.2020.9308409.

Dau, H. A., Bagnall, A., Kamgar, K., Yeh, C. C. M., Zhu, Y., Gharghabi, S., et al. (2019). The UCR time series archive. *IEEE/CAA Journal of Automatica Sinica*, *6*(6), 1293–1305. http://dx.doi.org/10.1109/jas.2019.1911747.

Dau, H. A., Silva, D. F., Petitjean, F., Forestier, G., Bagnall, A., Mueen, A., et al. (2018). Optimizing dynamic time warping's window width for time series data mining applications. *Data Mining and Knowledge Discovery*, *32*(4), 1074–1120. http://dx.doi.org/10.1007/s10618-018-0565-y.

del Campo, F. A., Neri, M. C. G., Villegas, O. O. V., Sánchez, V. G. C., de Jesús Ochoa Domínguez, H., & Jiménez, V. G. (2021). Auto-adaptive multilayer perceptron for univariate time series classification. *Expert Systems with Applications*, *181*, Article 115147. http://dx.doi.org/10.1016/j.eswa.2021.115147.

Dempster, A., Petitjean, F., & Webb, G. I. (2020). ROCKET: exceptionally fast and accurate time series classification using random convolutional kernels. *Data Mining and Knowledge Discovery*, *34*(5), 1454–1495. http://dx.doi.org/10.1007/s10618-020-00701-z.

Dempster, A., Schmidt, D. F., & Webb, G. I. (2021). Minirocket: A very fast (almost) deterministic transform for time series classification. In *Proceedings of the 27th ACM SIGKDD conference on knowledge discovery and data mining* (pp. 248–257). ACM, http://dx.doi.org/10.1145/3447548.3467231.

Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, *239*, 142–153. http://dx.doi.org/10.1016/j.ins.2013.02.030.

Džeroski, S., & Ženko, B. (2004). Is combining classifiers with stacking better than selecting the best one? *Machine Learning*, *54*(3), 255–273. http://dx.doi.org/10.1023/b:mach.0000015881.36452.6e.

Fan, C., Peng, Y., Peng, S., Zhang, H., Wu, Y., & Kwong, S. (2021). Detection of train driver fatigue and distraction based on forehead EEG: A time-series ensemble learning method. *IEEE Transactions on Intelligent Transportation Systems*, 1–11. http://dx.doi.org/10.1109/tits.2021.3125737.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P. A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, *33*(4), 917–963. http://dx.doi.org/10.1007/s10618-019-00619-1.

Fawaz, H. I., Lucas, B., Forestier, G., Pelletier, C., Schmidt, D. F., Weber, J., et al. (2020). InceptionTime: Finding AlexNet for time series classification. *Data Mining and Knowledge Discovery*, *34*(6), 1936–1962. http://dx.doi.org/10.1007/s10618-020-00710-y.

Fenton, N., & Neil, M. (2018). Decision analysis, decision trees, value of information analysis, and sensitivity analysis. In *Risk assessment and decision analysis with Bayesian networks* (pp. 347–369). Chapman and Hall/CRC, http://dx.doi.org/10.1201/b21982-11.

Feremans, L., Cule, B., & Goethals, B. (2022). PETSC: pattern-based embedding for time series classification. *Data Mining and Knowledge Discovery*, *36*(3), 1015–1061. http://dx.doi.org/10.1007/s10618-022-00822-7.

Forechi, A., Souza, A. F. D., Badue, C., & Oliveira-Santos, T. (2016). Sequential appearance-based global localization using an ensemble of kNN-DTW classifiers. In *2016 international joint conference on neural networks* (pp. 2782–2789). IEEE, http://dx.doi.org/10.1109/ijcnn.2016.7727550.

Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. *Journal-Japanese Society for Artificial Intelligence*, *14*(771–780), 1612.

Friedman, J. H. (2002). Stochastic gradient boosting. *Computational Statistics & Data Analysis*, *38*(4), 367–378. http://dx.doi.org/10.1016/s0167-9473(01)00065-2.

Galicia, A., Talavera-Llames, R., Troncoso, A., Koprinska, I., & Martínez-Álvarez, F. (2019). Multi-step forecasting for big data time series based on ensemble learning. *Knowledge-Based Systems*, *163*, 830–841. http://dx.doi.org/10.1016/j.knosys.2018.10.009.

Gao, Z. K., Cai, Q., Yang, Y. X., Dang, W. D., & Zhang, S. S. (2016). Multiscale limited penetrable horizontal visibility graph for analyzing nonlinear time series. *Scientific Reports*, *6*(1), http://dx.doi.org/10.1038/srep35622.

Gao, Y., & Yu, D. (2020). Total variation on horizontal visibility graph and its application to rolling bearing fault diagnosis. *Mechanism and Machine Theory*, *147*, Article 103768. http://dx.doi.org/10.1016/j.mechmachtheory.2019.103768.

Geurts, P. (2001). Pattern extraction for time series classification. In *European conference on principles of data mining and knowledge discovery* (pp. 115–127). Springer.

Geurts, P., Ernst, D., & Wehenkel, L. (2006). Extremely randomized trees. *Machine Learning*, *63*(1), 3–42. http://dx.doi.org/10.1007/s10994-006-6226-1.

Górecki, T., & Łuczak, M. (2014). Non-isometric transforms in time series classification using DTW. *Knowledge-Based Systems*, *61*, 98–108. http://dx.doi.org/10.1016/j.knosys.2014.02.011.

Gweon, H., & Yu, H. (2021). A nearest neighbor-based active learning method and its application to time series classification. *Pattern Recognition Letters*, *146*, 230–236. http://dx.doi.org/10.1016/j.patrec.2021.03.016.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *2016 IEEE conference on computer vision and pattern recognition* (pp. 770–778). IEEE, http://dx.doi.org/10.1109/cvpr.2016.90.

Hills, J., Lines, J., Baranauskas, E., Mapp, J., & Bagnall, A. (2013). Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, *28*(4), 851–881. http://dx.doi.org/10.1007/s10618-013-0322-1.

Iacobello, G., Scarsoglio, S., & Ridolfi, L. (2018). Visibility graph analysis of wall turbulence time-series. *Physics Letters. A*, *382*(1), 1–11. http://dx.doi.org/10.1016/j.physleta.2017.10.027.

Ji, C., Liu, S., Yang, C., Pan, L., Wu, L., & Meng, X. (2018). A shapelet selection algorithm for time series classification: New directions. *Procedia Computer Science*, *129*, 461–467. http://dx.doi.org/10.1016/j.procs.2018.03.025.

Ji, C., Zhao, C., Liu, S., Yang, C., Pan, L., Wu, L., et al. (2019). A fast shapelet selection algorithm for time series classification. *Computer Networks*, *148*, 231–240. http://dx.doi.org/10.1016/j.comnet.2018.11.031.

Jin, L.-p., & Dong, J. (2016). Ensemble deep learning for biomedical time series classification. *Computational Intelligence and Neuroscience*, *2016*, 1–13. http://dx.doi.org/10.1155/2016/6212684.

Kartha, M. J., Reshi, B. A., Walke, P. S., & Dastan, D. (2022). Morphological study of thin films: Simulation and experimental insights using horizontal visibility graph. *Ceramics International*, *48*(4), 5066–5074. http://dx.doi.org/10.1016/j.ceramint.2021.11.044.

Keogh, E., Chakrabarti, K., Pazzani, M., & Mehrotra, S. (2001). Dimensionality reduction for fast similarity search in large time series databases. *Knowledge and Information Systems*, *3*(3), 263–286. http://dx.doi.org/10.1007/pl00011669.

Keogh, E. J., & Pazzani, M. J. (2000). Scaling up dynamic time warping for datamining applications. In *Proceedings of the sixth ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 285–289). ACM Press, http://dx.doi.org/10.1145/347090.347153.

Lacasa, L., Luque, B., Ballesteros, F., Luque, J., & Nuño, J. C. (2008). From time series to complex networks: The visibility graph. *Proceedings of the National Academy of Sciences*, *105*(13), 4972–4975. http://dx.doi.org/10.1073/pnas.0709247105.

Lahreche, A., & Boucheham, B. (2021). A fast and accurate similarity measure for long time series classification based on local extrema and dynamic time warping. *Expert Systems with Applications*, *168*, Article 114374. http://dx.doi.org/10.1016/j.eswa.2020.114374.

Lan, X., Mo, H., Chen, S., Liu, Q., & Deng, Y. (2015). Fast transformation from time series to visibility graphs. *Chaos. An Interdisciplinary Journal of Nonlinear Science*, *25*(8), Article 083105. http://dx.doi.org/10.1063/1.4927835.

Large, J., Bagnall, A., Malinowski, S., & Tavenard, R. (2019). On time series classification with dictionary-based classifiers. *Intelligent Data Analysis*, *23*(5), 1073–1089. http://dx.doi.org/10.3233/ida-184333.

Laub, J., Roth, V., Buhmann, J. M., & Mller, K. R. (2006). On the information and representation of non-Euclidean pairwise data. *Pattern Recognition*, *39*(10), 1815–1826. http://dx.doi.org/10.1016/j.patcog.2006.04.016.

Lele, S., & Richtsmeier, J. T. (1991). Euclidean distance matrix analysis: A coordinate-free approach for comparing biological shapes using landmark data. *American Journal of Physical Anthropology*, *86*(3), 415–427. http://dx.doi.org/10.1002/ajpa.1330860307.

Li, H., Jia, R., & Wan, X. (2022). Time series classification based on complex network. *Expert Systems with Applications*, *194*, Article 116502. http://dx.doi.org/10.1016/j.eswa.2022.116502.

Li, D., Lin, J., Bissyande, T. F. D. A., Klein, J., & Le Traon, Y. (2018). Extracting statistical graph features for accurate and efficient time series classification. In *21st international conference on extending database technology* (pp. 1–12).

Liang, Z., & Wang, H. (2021). Efficient class-specific shapelets learning for interpretable time series classification. *Information Sciences*, *570*, 428–450. http://dx.doi.org/10.1016/j.ins.2021.03.063.

Lin, J., Khade, R., & Li, Y. (2012). Rotation-invariant similarity in time series using bag-of-patterns representation. *Journal of Intelligent Information Systems*, *39*(2), 287–315. http://dx.doi.org/10.1007/s10844-012-0196-5.

Lines, J., & Bagnall, A. (2014). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, *29*(3), 565–592. http://dx.doi.org/10.1007/s10618-014-0361-2.

Lines, J., Taylor, S., & Bagnall, A. (2016). HIVE-COTE: The hierarchical vote collective of transformation-based ensembles for time series classification. In *2016 IEEE 16th international conference on data mining* (pp. 1041–1046). IEEE, http://dx.doi.org/10.1109/icdm.2016.0133.

Lubba, C. H., Sethi, S. S., Knaute, P., Schultz, S. R., Fulcher, B. D., & Jones, N. S. (2019). catch22: CAnonical Time-series CHaracteristics. *Data Mining and Knowledge Discovery*, *33*(6), 1821–1852. http://dx.doi.org/10.1007/s10618-019-00647-x.

Lucas, B., Shifaz, A., Pelletier, C., O'Neill, L., Zaidi, N., Goethals, B., et al. (2019). Proximity forest: an effective and scalable distance-based classifier for time series. *Data Mining and Knowledge Discovery*, *33*(3), 607–635. http://dx.doi.org/10.1007/s10618-019-00617-3.

Luque, B., Lacasa, L., Ballesteros, F., & Luque, J. (2009). Horizontal visibility graphs: Exact results for random time series. *Physical Review E*, *80*(4), http://dx.doi.org/10.1103/physreve.80.046103.

Ma, Q., Zheng, Z., Zhuang, W., Chen, E., Wei, J., & Wang, J. (2021). Echo Memory-Augmented Network for time series classification. *Neural Networks*, *133*, 177–192. http://dx.doi.org/10.1016/j.neunet.2020.10.015.

Mahdavinejad, M. S., Rezvan, M., Barekatain, M., Adibi, P., Barnaghi, P., & Sheth, A. P. (2018). Machine learning for internet of things data analysis: a survey. *Digital Communications and Networks*, *4*(3), 161–175. http://dx.doi.org/10.1016/j.dcan.2017.10.002.

Middlehurst, M., Vickers, W., & Bagnall, A. (2019). Scalable dictionary classifiers for time series classification. In *Intelligent data engineering and automated learning* (pp. 11–19). Springer International Publishing, http://dx.doi.org/10.1007/978-3-030-33607-3_2.

Papadopoulos, S., & Karakatsanis, I. (2015). Short-term electricity load forecasting using time series and ensemble learning methods. In *2015 IEEE power and energy conference at Illinois* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/peci.2015.7064913.

Qiu, X., Zhang, L., Ren, Y., Suganthan, P., & Amaratunga, G. (2014). Ensemble deep learning for regression and time series forecasting. In *2014 IEEE symposium on computational intelligence in ensemble learning* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/ciel.2014.7015739.

Rincy, T. N., & Gupta, R. (2020). Ensemble learning techniques and its efficiency in machine learning: A survey. In *2nd international conference on data, engineering and applications* (pp. 1–6). IEEE, http://dx.doi.org/10.1109/idea49133.2020.9170675.

Ryabko, D., & Mary, J. (2012). Reducing statistical time-series problems to binary classification. *Advances in Neural Information Processing Systems*, *3*.

Sagi, O., & Rokach, L. (2018). Ensemble learning: A survey. *WIREs Data Mining and Knowledge Discovery*, *8*(4), Article e1249. http://dx.doi.org/10.1002/widm.1249, arXiv:https://wires.onlinelibrary.wiley.com/doi/pdf/10.1002/widm.1249.

Schäfer, P. (2014). The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, *29*(6), 1505–1530. http://dx.doi.org/10.1007/s10618-014-0377-7.

Schäfer, P., & Högqvist, M. (2012). SFA: a symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the 15th international conference on extending database technology* (pp. 516–527). ACM Press, http://dx.doi.org/10.1145/2247596.2247656.

Schäfer, P., & Leser, U. (2017). Fast and accurate time series classification with WEASEL. In *Proceedings of the 2017 ACM on conference on information and knowledge management* (pp. 637–646). ACM, http://dx.doi.org/10.1145/3132847.3132980.

Shifaz, A., Pelletier, C., Petitjean, F., & Webb, G. I. (2020). TS-CHIEF: a scalable and accurate forest algorithm for time series classification. *Data Mining and Knowledge Discovery*, *34*(3), 742–775. http://dx.doi.org/10.1007/s10618-020-00679-8.

Stefan, A., Athitsos, V., & Das, G. (2013). The move-split-merge metric for time series. *IEEE Transactions on Knowledge and Data Engineering*, *25*(6), 1425–1438. http://dx.doi.org/10.1109/tkde.2012.88.

Sun, Y., Li, J., Liu, J., Sun, B., & Chow, C. (2014). An improvement of symbolic aggregate approximation distance measure for time series. *Neurocomputing*, *138*, 189–198. http://dx.doi.org/10.1016/j.neucom.2014.01.045.

Tan, C. W., Dempster, A., Bergmeir, C., & Webb, G. I. (2022). MultiRocket: multiple pooling operators and transformations for fast and effective time series classification. *Data Mining and Knowledge Discovery*, http://dx.doi.org/10.1007/s10618-022-00844-1.

Wang, B., Jiang, T., Zhou, X., Ma, B., Zhao, F., & Wang, Y. (2020). Time-series classification based on fusion features of sequence and visualization. *Applied Sciences*, *10*(12), 4124. http://dx.doi.org/10.3390/app10124124.

Wang, T., Liu, Z., Zhang, T., Hussain, S. F., Waqas, M., & Li, Y. (2022). Adaptive feature fusion for time series classification. *Knowledge-Based Systems*, *243*, Article 108459. http://dx.doi.org/10.1016/j.knosys.2022.108459.

Wang, H., Wu, Q. J., Wang, D., Xin, J., Yang, Y., & Yu, K. (2021). Echo state network with a global reversible autoencoder for time series classification. *Information Sciences*, *570*, 744–768. http://dx.doi.org/10.1016/j.ins.2021.04.074.

Wang, Z., Yan, W., & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *2017 international joint conference on neural networks* (pp. 1578–1585). IEEE, http://dx.doi.org/10.1109/ijcnn.2017.7966039.

Wei, W. W. (2013). *Time series analysis*. Oxford University Press, http://dx.doi.org/10.1093/oxfordhb/9780199934898.013.0022.

Wu, Y., & Gao, J. (2018). Adaboost-based long short-term memory ensemble learning approach for financial time series forecasting. *Current Science*, *115*(1), 159. http://dx.doi.org/10.18520/cs/v115/i1/159-165.

Wu, S., Wang, X., Liang, M., & Wu, D. (2021). PFC: A novel perceptual features-based framework for time series classification. *Entropy*, *23*(8), 1059. http://dx.doi.org/10.3390/e23081059.

Xi, X., Keogh, E., Shelton, C., Wei, L., & Ratanamahatana, C. A. (2006). Fast time series classification using numerosity reduction. In *Proceedings of the 23rd international conference on machine learning* (pp. 1033–1040). ACM Press, http://dx.doi.org/10.1145/1143844.1143974.

Xu, P., Zhang, R., & Deng, Y. (2018). A novel visibility graph transformation of time series into weighted networks. *Chaos, Solitons & Fractals*, *117*, 201–208. http://dx.doi.org/10.1016/j.chaos.2018.07.039.

Yu, D., Yu, X., Hu, Q., Liu, J., & Wu, A. (2011). Dynamic time warping constraint learning for large margin nearest neighbor classification. *Information Sciences*, *181*(13), 2787–2796. http://dx.doi.org/10.1016/j.ins.2011.03.001.

Zhang, H., Dong, Y., & Xu, D. (2020). Entropy-based symbolic aggregate approximation representation method for time series. In *2020 IEEE 9th joint international information technology and artificial intelligence conference* (pp. 905–909). IEEE, http://dx.doi.org/10.1109/itaic49862.2020.9339021.

Zhang, Y., Ren, G., Liu, X., Gao, G., & Zhu, M. (2021). Ensemble learning-based modeling and short-term forecasting algorithm for time series with small sample. *Engineering Reports*, *4*(5), http://dx.doi.org/10.1002/eng2.12486.

Zhong, Y., Chen, W., Wang, Z., Chen, Y., Wang, K., Li, Y., et al. (2020). HELAD: A novel network anomaly detection model based on heterogeneous ensemble learning. *Computer Networks*, *169*, Article 107049. http://dx.doi.org/10.1016/j.comnet.2019.107049.

Zhou, C., Ding, L., Zhou, Y., & Skibniewski, M. J. (2019). Visibility graph analysis on time series of shield tunneling parameters based on complex network theory. *Tunnelling and Underground Space Technology*, *89*, 10–24. http://dx.doi.org/10.1016/j.tust.2019.03.019.