A novel time series classification for multivariate data using improved deep belief-recurrent neural network with optimal dynamic time warping

Bukya M. Babu and B. Sandhya

1 Department of CSE, UCE, Osmania University, Hyderabad, Telangana, India.
2 Department of Computer Science and Engineering, Maturi Venkata Subba Rao (MVSR) Engineering College, Hyderabad, Telangana, India.

Abstract. In the past ten years, data from time series extraction has attracted a lot of attention. Several methods have concentrated on classification problems, where the objective is to identify the labelling of a test period, given labelled training data. Feature-based and Instance-based methods are the two fundamental groups into which time series categorization methodologies may be divided. To categorize time series data, instance-based techniques use similarity data in a nearest-neighbor context. While methods in this category deliver reliable findings, their efficacy suffers when dealing with lengthy and noisy time series. Feature-based approaches, on the other hand, extract characteristics to address the shortcomings of instance-based methods; nevertheless, these approaches use predetermined features and might not be effective in all classification issues. This paper seeks to introduce a novel deep learning-based Optimal Dynamic Time Warping (ODTW) paradigm for multimodal time series data categorization. This model covers several phases. At initial stage, the standard data is gathered from standard public source. Secondly, ODTW is proposed, where the parameters are optimized by Random Opposition Billiards-Inspired Optimization (RO-BIO) for extracting the most essential information. Finally, the classification is carried out through “Deep Belief-Recurrent Neural Network (DB-RNN)”. Finally, the extracted deep features are given to the optimized RNN for attaining the final classified results. The simulation results have resulted in superior classification performance in terms of standard performance measures.

Keywords. Time Series Classification; Multivariate Data; Optimal Dynamic Time Warping; Random Opposition Billiards-Inspired Optimization; Deep Belief-Recurrent Neural Network

1 Introduction

Time series classification is a fascinating and essential area of machine learning and data analysis that deals with making predictions or categorizations based on ordered sequences of

------------------------------------------------------------------
© The Authors, published by EDP Sciences. This is an open access article distributed under the terms of the Creative Commons Attribution License 4.0 (https://creativecommons.org/licenses/by/4.0/).
data points [9]. Unlike traditional classification tasks where each data point is treated independently, time series classification takes into account the temporal aspect of the data, allowing us to capture patterns, trends, and dependencies that evolve over time [10]. The classification of time series is a specialized branch of deep learning that focuses on categorizing time-varying data points into different classes or categories [11]. This can have applications in various fields such as finance, healthcare, manufacturing, and more. Deep learning techniques, especially RNNs and CNNs have proven to be highly effective for solving time series classification tasks [12]. Time series data consists of sequences of data points recorded over time. Each data point typically has one or more features associated with it, and the order of these data points is crucial for analysis [13]. It's essential to understand the characteristics of your time series data, such as seasonality, trends, and noise [14].

While deep learning has shown significant promise in addressing time series classification tasks [15], it also comes with several limitations and challenges that researchers and practitioners need to be aware of. Deep learning models, especially those with a large number of parameters, require substantial amounts of data for effective training. In cases where labeled time series data is scarce, the performance of deep learning models can be compromised [16]. Additionally, if the data is noisy or contains outliers, it can adversely affect the model's ability to learn meaningful patterns [17]. Deep learning models can be computationally intensive, particularly for complex architectures like deep recurrent or convolutional networks [18]. Training such models can demand significant computational resources, including powerful GPUs or specialized hardware [19]. Deep learning models involve multiple hyperparameters, such as batch size, learning rate and network architecture. Tuning these hyperparameters for optimal performance can be time-consuming and require expertise [20]. Some deep learning models struggle to focus on relevant time steps within a sequence. This can result in the model giving equal attention to all time steps, even when only certain parts of the sequence are relevant for classification. Despite these limitations, researchers and practitioners continue to develop methods to address these challenges and improve the performance of deep learning models for time series classification. It's important to carefully consider these limitations and tailor your approach accordingly based on the specific characteristics of your data and problem [21].

Time series classification using deep learning involves a variety of techniques that are designed to capture temporal dependencies and patterns within sequences of data. RNNs are a class of neural networks designed to handle sequences by maintaining a hidden state that captures temporal dependencies [22]. They are suitable for capturing sequential patterns in time series data. “Gated Recurrent Unit (GRU) and Long Short-Term Memory (LSTM)” are popular RNN variants that help mitigate the vanishing gradient problem and can capture long-range dependencies [23]. Hybrid models combine the strengths of RNNs and CNNs to capture both local and temporal patterns. Combining predictions from multiple models can enhance the overall performance and robustness of time series classification models [24]. Autoencoders are neural networks designed to learn efficient representations of input data. These techniques can be adapted and combined to address the specific challenges of time series classification tasks [25].

The primary importance of the advanced model is elucidated as follows.

• To develop a time series classification based on deep learning network; it is used to analyze and classify data points that are collected over time and it is useful in finance, healthcare, manufacturing, and more.
• To gather standard data from public sources, a new optimization technique, the proposed algorithm RO-BIO is employed to optimize the extraction of essential features from the data. Specifically, the algorithm is applied to the ODTW distance calculation to extract
important information and design the DB-RNN model, this combines the DBN and RNN for classification tasks

- The simulation results indicate that the suggested model demonstrates improved classification performance compared to standard methods. This performance improvement is measured using standard performance metrics.
- The following section labels appear to be fitting for the upcoming content. Section II presents an exploration of research related to time series classification methods. Section III delves into the application of ODTW and a hybrid deep learning technique for multivariate time series classification. In Section IV, the introduction of the innovative RO-BIO approach is discussed, focusing on its use for optimal dynamic time warping in classification. The utilization of DB-RNN for time series classification is covered in Section V. The methodology, execution process, and results of the simulations are detailed in Section VI. The achieved results of the primary objective are documented in Section VII.

2 Existing works

2.1 Related works

In 2023, M. H. Tahan et al. [1] have incorporated temporal discretization as a preprocessing step for time series data, seamlessly integrated it within DNN. These novel models were structured in two main segments: model training and temporal discretization. In the initial phase, the focused lies on discretization and, to some extent, the selection of fundamental features. The subsequent phase delved into the task of pinpointing more precise features and facilitating classification. To accomplished this, a dual-pronged approach was employed. The first evaluated the quality of discretization achieved, while the second gauged the classification accuracy. Through empirical evaluation across a set of 20 benchmark multivariate time series datasets, these proposed methods showcase superior accuracy compared to existing state-of-the-art techniques.

In 2018, A. Gharehbaghi and M. Lindén [2] have reported a pioneering validation approach was introduced to comprehensively assessed structural risk, encompassing both quantitative and qualitative dimensions. The influenced of “Dynamic Temporal Graph Neural Network (DTGNN)” on classifier performance was substantiated through rigorous statistical validation employing repeated random subsampling across distinct sets of Continuous Time Series (CTS) data originated from various medical applications. This validation protocol was extended to encompassed four distinct medical databases, each comprised specific signal types. These databases consisted of 108 electroencephalogram signal recordings, 90 electromyogram signal recordings, 130 heart sound signal recordings, and 50 respiratory sound signal recordings. The outcomes of these meticulous statistical validations unequivocally demonstrated that the utilization of DTGNN leads to significant enhancement in classification performance.

In 2023, M. H. Tahan et al. [3] have recommended three comprehensive end-to-end deep learning models was introduced in this work, denoted as FCN-DISC, ALSTM-FCN-DISC and LSTM-FCN-DISC. These models synergistically leverage the advantages of both deep network architecture and temporal discretization. The primary objective of these models was to intelligently selected pertinent values within input time-series data, optimizing their impact on model training through the integration of temporal discretization within deep network structures. This novel model was designed around the utilization of dual loss functions. These
loss functions collaboratively contributed to the creation of discretized time series representations while concurrently refining network weights. To this end, a novel loss function dedicated to the discretization process was introduced; in addition to the established cross-entropy loss. Empirical evaluations conducted on univariate time series classification datasets underscore the efficacy of the proposed models.

In 2022, Ran Liu et al. [4] have encompassed three sequential phases: time-series imaging, classification and parameter optimization. In the parameter optimization phase, a novel optimization technique was introduced to fine-tune the reconstruction parameters. This optimization was performed with the objective of maximizing the image resolution achieved during the conversion process. During the time-series imaging phase, the temporal data underwent a transformation into a phase space configuration, leveraging the parameters that had been optimized beforehand. Notably, this methodology treated the time series' trajectory matrix as an image directly, eliminating the necessity for a projection of the trajectory into phase space. This particular approach efficiently circumvented the common issue of information loss that often accompanies such projections. Concluding this process, the classification phase involved subjecting the images derived from the time series data to classification using a DCNN. This classification process capitalized on the distinctive features extracted from the images, thus enabled accurate and effective categorization.

In 2019, Liu et al. [5] have introduced the architecture of MVCNN was specifically designed to capture both the multivariate temporal dependencies and nature of the data encoded by lagged features. The effectiveness of this proposed approach was demonstrated through its application to the Prognostics and Health Management (PHM) 2015 challenge dataset. A comprehensive comparison was conducted against various other algorithms commonly used for this type of data analysis. The evaluation metric employed was the prediction score, a key assessment criterion in the PHM Society 2015 data challenge. Beyond the performance evaluation, the study delved into a detailed analysis of the MVCNN approach, providing insights into its inner workings and shedding light on its efficacy in handling multivariate time series classification tasks.

In 2023, Y. Wu et al. [6] have recommended the proposed model, named the Multivariate time series classification Convolutional Transformer Network (MCTNet), leveraged a unique blend of Transformer encoders and convolutional structures. This hybrid architecture capitalized on the distinct strengths of both CNN and self-attention mechanisms to more effectively capture intricate patterns within multivariate time series data. Recognizing the data-intensive nature of transformers, the model seeked to overcome this limitation by fusing the inductive biases inherent in CNN. This was achieved through an innovative approach where initial features was extracted via convolutional layers. Additionally, a novel architecture called the “Both Squeeze and Excitation Convolutional Encoder (BC-Encoder)” was introduced to further enhanced the information extraction process.

In 2020, Bin Qian et al. [7] have introduced an innovative dynamic multi-scale-CNN was designed to capture a wide range of diverse multi-scale feature patterns inherent in each time series data instance. The core innovation of this architecture revolved around crafting a flexible filter generator capable of producing filters of varying lengths, tailored to the specific input time series. Central to maintaining model differentiability was the incorporation of learnable soft masks, which regulated the lengths of these adaptable filters. This mechanism facilitated the extraction of feature representations across various time scales, enabled by the dynamic filters. To pinpoint the most informative local patterns within these feature representations, a max-over-time pooling mechanism was deployed. This process selectively
highlighted crucial details. Ultimately, a fully connected layer, culminating in a softmax output, was employed. This layer computed a probability distribution across distinct classes, leading to the final classification decision.

In 2017, Zhao et al. [8] have suggested a CNN architecture was introduced, tailored specifically for time series classification tasks. In contrast to traditional feature-centric classification methods, this CNN framework was designed to autonomously discern and extract pertinent internal structures from input time series data. This was achieved through the utilization of convolutional and pooling operations, enabling the automatic generation of deep features. The superiority of the proposed method was evidenced by its impressive classification accuracy and robustness against noise, signifying its potential as an advanced tool for accurate and reliable time series classification tasks.

2.2 Research Gaps and Challenges

Time series classification involves assigning labels or categories to sequences of data points ordered over time, and it requires specialized techniques to handle the temporal nature of the data effectively. Several traditional techniques like DNN [1] is especially valuable when dealing with complex temporal patterns and time series data often exhibit hierarchical patterns at different time scales but it can be crucial for achieving good performance in time series classification tasks. DTGNN [2] is designed to specifically handle time series data, it might offer advantages in capturing temporal dependencies and dynamics in the data, which is crucial for accurate time series classification but the adoption of a new architecture can be a limitation if it's not well-supported or integrated with existing machine learning environments. DNN [3] can handle such irregularities by using sequence-to-sequence architectures or attention mechanisms and it can have high dimensionality due to the time dimension but it can capture patterns in sequential data, they might lack the ability to deeply understand the context or semantics of the data, leading to potential misclassifications. DCNN [4] can have high dimensionality due to the time dimension but it can process this high-dimensional data efficiently, learning meaningful representations that capture temporal relationships and it can suffer from vanishing or exploding gradient problems when dealing with very long sequences, impacting training stability. CNN [5] can handle time series of varying lengths without requiring excessive pre-processing and it could be beneficial for real-world datasets where time series have different durations but it often come with increased complexity, which can lead to longer training times and higher computational requirements. MCTNet [6] can be advantageous for identifying important features and patterns within multivariate time series data but the effectiveness of MCTNet might vary across different domains and types of multivariate time series data. The architecture could be specialized to certain types of patterns and less effective for others. CNN [7] has the ability to autonomously acquire pertinent features directly from raw time series data, thereby diminishing the requirement for laborious and domain-constrained manual feature engineering. Moreover, its applicability extends seamlessly to the realm of multivariate time series data through the application of convolutional operations spanning diverse dimensions. This approach adeptly captures intricate interactions among multiple variables. However, it's important to note that for tasks where comprehending overarching trends holds significance, this methodology might impose limitations. CNN [8] can be parallelized across different feature maps, making them computationally efficient and suitable for modern hardware like GPUs but it can be prone to overfitting, especially when dealing with small datasets. Regularization techniques need to be applied to mitigate this. Thus, these limitation helps to improved the new time series classification based on deep learning method.
Table 1. Features and challenges of existing time series classification based on deep learning method

<table>
<thead>
<tr>
<th>Author [citation]</th>
<th>Methodology</th>
<th>Features</th>
<th>Challenges</th>
</tr>
</thead>
</table>
| M. H. Tahan et al. [1] | DNN | • This is especially valuable when dealing with complex temporal patterns.  
• Time series data often exhibit hierarchical patterns at different time scales. | • It can be crucial for achieving good performance in time series classification tasks. |
| A. Gharehbaghi and M. Lindén et al. [2] | DTGNN | • It is designed to specifically handle time series data, it might offer advantages in capturing temporal dependencies and dynamics in the data, which is crucial for accurate time series classification. | • The adoption of a new architecture can be a limitation if it's not well-supported or integrated with existing machine learning environments. |
| M. H. Tahan et al. [3] | DNN | • It DNNs can handle such irregularities by using sequence-to-sequence architectures or attention mechanisms.  
• It can have high dimensionality due to the time dimension. | • It can capture patterns in sequential data; they might lack the ability to deeply understand the context or semantics of the data, leading to potential misclassifications. |
| Ran Liu et al. [4] | DCNN | • It can have high dimensionality due to the time dimension.  
• It can process this high-dimensional data efficiently, learning meaningful representations that capture temporal relationships. | • It can suffer from vanishing or exploding gradient problems when dealing with very long sequences, impacting training stability. |
<table>
<thead>
<tr>
<th>Authors</th>
<th>Method</th>
<th>Advantages</th>
<th>Limitations</th>
</tr>
</thead>
</table>
| C. -L. Liu et al. [5] | CNN    | • It can handle time series of varying lengths without requiring excessive pre-processing.  
• It could be beneficial for real-world datasets where time series have different durations. | • It often comes with increased complexity, which can lead to longer training times and higher computational requirements. |
| Y. Wu et al. [6]   | MCTNet | • This can be advantageous for identifying important features and patterns within multivariate time series data.                                                                                       | • The effectiveness of MCTNet might vary across different domains and types of multivariate time series data. The architecture could be specialized to certain types of patterns and less effective for others |
| Bin Qian et al. [7] | CNN    | • It has the ability to autonomously acquire pertinent features directly from raw time series data, thereby diminishing the requirement for laborious and domain-constrained manual feature engineering.  
• Its applicability extends seamlessly to the realm of multivariate time series data through the application of convolutional operations spanning diverse dimensions. This approach adeptly captures intricate interactions among multiple variables. | • This can be limiting for tasks where understanding overall trends is crucial. |
It can be parallelized across different feature maps, making them computationally efficient and suitable for modern hardware like GPUs.

It can be prone to overfitting, especially when dealing with small datasets. Regularization techniques need to be applied to mitigate this.

3 Multivariate data for time series classification using optimal DTW and hybrid deep learning technique

3.1 Proposed Description of Time Series Classification

A time series classification system is a framework designed to analyze and categorize data points that are recorded sequentially over time. Time series data consists of observations made at specific time intervals, resulting in a sequence of data points that exhibit temporal dependencies. Time series classification involves assigning a label or category to each sequence based on its patterns, trends, or other features. Time series classification systems excel at capturing patterns and trends in data that evolve over time. This allows for a deeper understanding of how events unfold and change over sequential observations. Time series classification can be instrumental in early anomaly detection. By recognizing abnormal patterns as they emerge, this approach can prevent potentially harmful situations or events but imbalanced class distributions are common in time series classification tasks, where one class might be significantly more frequent than others. Handling class imbalance is crucial to avoid biased model performance. To enhance the performance of a time series classification system and overcome its inherent limitations, a proposed approach can be implemented. This approach aims to optimize various aspects of the system to achieve improved results. Fig 1 shows the pictorial presentation of the recommended RO-BIO based time series classification system.

![Diagram of the proposed RO-BIO based time series classification system]

Fig. 1. The pictorial presentation of the suggested RO-BIO based time series classification system
The proposed system consists of different phases, initially gather standard data from public sources, a new optimization technique, the proposed algorithm RO-BIO is employed to optimize the extraction of essential features from the data. Specifically, the algorithm is applied to the DTW distance calculation to extract important information. To design the DB-RNN model, this combines the DBN and RNN for classification tasks. The simulation results indicate that the proposed model demonstrates improved classification performance compared to standard methods. This performance improvement is measured using standard performance metrics.

3.2 Implemented Dataset Details

In this proposed time series classification system, the dataset is manually generated, which is described below. Dataset description (TSC/TSCl Data): The link is “https://www.timeseriesclassification.com/; Access Date 2023-08-10”. The TSC/TSCl dataset consists of classification problems involving both univariate and multivariate time series data. These problems are presented in three different formats: Weka ARFF, plain text files, and aeon format. Notably, the Weka ARFF format has a limitation when it comes to handling time series of unequal lengths. Therefore, the manually generated datasets are labelled with here , the term is the quantity of collected anomaly data.

4 Optimal dynamic time warping using novel ro-bio approach for classification

4.1 Random Opposition of BIO

In the recently established novel time series classification system model, a newly introduced RO-BIO algorithm plays a pivotal role. This algorithm is designed to enhance outcomes by addressing the limitations present in the conventional classical model. Inherently balances exploration and exploitation due to its simulation of billiard ball movements. This balance can help prevent premature convergence to local optima and encourage the algorithm to explore diverse regions of the search space but depending on the problem and the complexity of the optimization landscape, BIO [26] can be computationally intensive. Simulating the interactions between multiple “balls” and handling collisions can increase the algorithm's computational overhead. Consequently, through the formulation of the novel solution encoding process, the constraints of the existing BIO models are bypassed, leading to the introduction of the RO-BIO concept. This innovative approach culminates in the development of a freshly derived equation, presented as Eq. (1).

\[ \gamma = -t \frac{1}{Max_{iter}} \]

\[ \gamma \text{ is the random number among } [0 - 1]. \]

\[ \text{Also the variable } iter \text{ defines the maximum iteration. The proposed algorithm is described below. BIO belongs to the family of metaheuristic algorithms, which are problem-solving techniques that simulate natural processes, such as evolution, swarm behavior, and physical phenomena, to find optimal or near-optimal solutions to optimization problems. In the case of BIO, the algorithm is particularly focused on optimization tasks that involve continuous variables and non-convex search spaces. The core concept of the BIO algorithm lies in its emulation of the motion of billiard balls. Imagine the optimization problem as a multi-dimensional space, where each dimension corresponds to a variable to be optimized. Similar, 01161 (2024)MATEC Web of Conferences https://doi.org/10.1051/matecconf/202439201161 392 ICMED 2024 \]
Initialization:
The initial set of balls is scattered haphazardly throughout the search space as shown in Eq. (2)...

\[ K_{a,b}^{s} = V_{a_{b}} + \gamma V_{a_{b}} - V_{a_{b}} = \] ...(2)

Here, the initial state of the \( a \) variable for the \( b \) ball is established using \( s_{b} \). This sets the starting point for the variable. The range within which the \( a \) variable can vary is defined by \( \text{max}_{b} V_{a} \) and \( \text{min}_{b} V_{a} \), indicating its upper and lower limits.

Evaluation:
During this stage, both the positions of the balls and the locations of the pockets are assessed based on their performance according to the objective function. The objective function represents the criteria or goals that need to be optimized or achieved.

Selecting the Pockets:
In BOA algorithm, the pockets serve a dual purpose. They act as attractive points for the balls, enhancing the algorithm's capacity for exploitation. Moreover, they function as a repository, preserving the initial most exceptional solutions uncovered up to that point. This storage mechanism enhances the algorithm's effectiveness without imposing additional computational burdens. Consequently, this storage is continuously refreshed and replaced by the superior positions identified by the best-performing balls in every iteration.

Categorizing the Balls:
The balls are arranged in order of their fitness scores and subsequently divided into two equal factions: regular balls and cue balls. The superior half encompasses the regular balls while the latter section designates the cue balls. Each cue ball corresponds to a matching rank within the higher-tier group.

Allocating the pockets:
The proposed probability for pocket selection is as following Eq. (3):

\[ L_{s} = k \sum e^{(\beta \chi)} \]

Revitalizing Ball Positions:
The accuracy of the shot impacts these positions. As part of an effort to enhance exploitation capabilities, the degree of error diminishes as the search process unfolds. The ensuing Eq.(4) and Eq.(5) establishes the updated positions for the ordinary balls.

\[ K_{a,b}^{new} = \gamma \cdot \text{posy} L \cdot K_{a,b}^{old} - S_{a,b}^{m} \pm S_{a,b}^{m} a = \]
\[ LK = \frac{\text{Iter}}{\text{Max}_\text{iter}} \]

\[ \overrightarrow{K}_{a:b}^{\text{old}} = b_{th} \]

\[ \overrightarrow{K}_{a:b}^{\text{new}} = S_{s_{th}}^m \]

\[ m_{th} \]

\[ \overrightarrow{H}_n = \sqrt{b K_a^{\text{old}} K_a^{\text{new}}} \]

\[ K_{a+b}^{\text{new}} = \frac{B_{a+b}}{n} b_{th} + K_{a+b}^{\text{old}} \]

Escaping process:

\[ K_{a:b}^s = V_{a:b} + \gamma V_{a:b} - V_{a:b} \]

Algorithm 1: Proposed RO-BIO

Assume No. of balls \( A \)

No. of variables \( a \)

No. of pockets \( L \)

Escaping threshold \( IL \)

The random parameter \( \gamma \) is calculated by Eq. (1)

Do While

\[ \text{Iter} < \text{Max}_\text{iter} \]

Initialize \( A \) balls and \( L \) pockets by Eq.(2)

Assess the arrangement of the balls and pockets in relation to the target positions.

Revise the pocket database and population records.

Establish distinct groups for regular balls and the cue ball.

For each pair ball
Employ the roulette-wheel selection method to choose a target pocket.

**End**

Revise the position of the current standard ball using Eq. (3).
Compute the post-collision velocity of the standard ball using Eq. (4).
Calculate the post-collision velocity of the cue ball using Eq. (5).
Update the position of the current cue ball using Eq. (6).

**If** ($\gamma < IL$)

Generate a new set of randomized condition constraints and rectify the ball positions if they fall outside the specified range.

**Iter = Iter + 1**

**End**

Provide the optimal pocket as the final solution.

---

**Fig 2.** Flowchart for suggested RO-BIO approach
4.2 Traditional DTW

DTW is a widely used algorithm in the fields of signal processing, time series analysis, and pattern recognition. It's designed to measure the similarity between two sequences that might vary in speed or timing. The gathered data \( R_s^{po} \) is the input to this phase, unlike traditional distance measures that assume a linear relationship between data points, DTW takes into account the non-linear warping of time axes, allowing it to handle sequences with different lengths and temporal distortions.

The methodology of DTW [34] involves leveraging dynamic programming techniques to identify a spectrum of potential paths. Among these paths, the optimal one is chosen by minimizing the distance within the two time series. This distance is computed through a matrix of distances, where each matrix element signifies the cumulative distance derived from the smallest values among its three neighbouring components.

Let's consider two given time series sequences: \( D = D_1 \ldots D_n \) and \( a \cdot b \). In the initial step, we construct a matrix with dimensions \( a \times b \). Within this matrix, each element at position \( a \cdot b \) corresponds to the cumulative distance associated with the distance at that specific point, in addition to the minimum value among the three neighbouring elements. Here, \( a \) ranges from 1 to \( < a < n \), and \( b \) ranges from 1 to \( < b < m \). Mathematically, we can denote the \( a \cdot b \) element as following Eq.(9).

\[
e_{ab} = p_{ab} + \min\left\{a_{a-1 \cdot b-1}, e_{a-1 \cdot a}, e_{i \cdot b-1}\right\}
\]

Here, \( f_{ab} = u_a + w_b \) and \( e_{ab} \) is the matrix's \( a \cdot b \) component, and it's the total of the squared differences among \( a \cdot b \) and \( u_a + w_b \) and the least cumulative distances among \( a \cdot b \) element's three adjacent components, are called the \( a \cdot b \) component. The route that delivers the smallest total distance at \( n \cdot m \) is the one to choose next in order to find the best course of action. The distance is explained as Eq. (10).

\[
T_{DWT}[W \cdot U] = \sum_{k=1}^{k} \sqrt{\sum_{k=1}^{k} E_w R_s^{po}}
\]

In this scenario, the algorithm might encounter situations where multiple neighboring elements have the same value for the cumulative distance. As a result, when making a choice among these tied neighbors, the algorithm's decision might appear arbitrary. Despite this apparent arbitrariness, it's important to note that regardless of the chosen neighbor, the resulting optimal warping distance remains consistent. In essence, different optimal paths might be selected, but they ultimately yield the same warping distance. Finally obtained the extracted feature and the next process is detection, extracted feature is expressed as \( R_s^{ef} \).

4.3 Optimal DTW using Weights

DTW is a popular technique used to measure the similarity between two sequences that might be of different lengths and have varying speeds. Creating a distance matrix using DTW involves calculating the DTW distance between all pairs of sequences in a dataset. This
matrix provides a comprehensive view of the similarity or dissimilarity between each pair of sequences. Incorporating weights into the DTW algorithm increases its computational complexity. The additional multiplication by weights during distance calculations and accumulation can slow down the algorithm, especially for long sequences or large datasets. To lessen these errors, using proposed algorithm and optimizing parameters like weight in DWT. This helps to improve the correlation coefficient. The objective function of the optimal DTW using weights is mathematically described in Eq. (11).

\[ F_o = \text{Weight}_{DWT} \cdot C_{coee} \]  

(11)

From the form, the term \( \text{Weight}_{DWT} \) shows the weight in DWT and the range is \([1 - 10]\). Also the term \( C_{coee} \) defines the correlation coefficient. The correlation coefficient is a statistical measure that quantifies the degree of linear relationship between two variables. It provides insights into how changes in one variable are associated with changes in another variable. The mathematical formulation of \( C_{coee} \) is shown in Eq.(12).

\[ C_{coee} = \frac{\sum_{i=1}^{k} (a_i - \bar{a}) (b_i - \bar{b})}{\sqrt{\sum_{i=1}^{k} (a_i - \bar{a})^2 (b_i - \bar{b})^2}} \]  

(12)

Here, the term \( a_i \) and \( \bar{a} \) defines the \( i_{th} \) value of the first variable and mean value. Also the variable \( b_i \) and \( \bar{b} \) defines the \( i_{th} \) value of the second variable and mean value.

5 Time series classification using Deep Belief-Recurrent Neural Network

5.1 Deep Belief Network

A DBN (Deep Belief Network), referenced in [32], constitutes a variant of artificial neural networks characterized by a multi-layer architecture comprised of interconnected units. Its primary objective is to acquire hierarchical understandings of data. Within the realm of generative models, DBNs hold a pivotal role, rendering them valuable for tasks rooted in unsupervised learning such as feature acquisition, dimensionality reduction, and generative modeling.

DBNs consist of two principal layer types: the visible layer and the hidden layers. During this phase, the extracted features function \( R_s \) as inputs. The visible layer stands as a representation of the input data, while the hidden layers progressively capture more intricate and elevated features or representations. Each layer maintains complete connectivity with the layers above and below, signifying that every unit within a given layer establishes connections with all units in adjacent layers.

From a decoding perspective, one can conceptualize a DBN as akin to a multi-layer Perceptron containing numerous layers. The input signal undergoes sequential processing through each layer, following the procedure defined by Eq. (1), until reaching the ultimate layer. In this final layer, the output is subsequently subjected to a transformation that yields
a multinomial distribution, achieved through the application of the softmax operation as shown in Eq. (13) and Eq. (14).

\[ m^k_p = \sigma \left( \sum_{i=1}^{v} u_i v_i + s_i \right) R_s^{ef} \]  

(13)

\[ m = l = k \quad k \quad \Theta = \frac{m^k_p}{v^k} R_s^{ef} \]  

(14)

In the context where "l = k" signifies the classification of the input into the sixth class, and \( u_i \) represents the weight connecting the hidden \( k_{p,i} \) in the final layer to the class label \( k \).

As our foundational approach, we employ the traditional frame-level criterion for the training of DBNs. Specifically, we adhere to the methodology introduced in for the optimization of DBN weights. This involves an initial training phase wherein a series of Restricted Boltzmann Machines (RBMs) are trained in a generative manner. Subsequently, a fine-tuning process is conducted, where all parameters are adjusted collectively through the utilization of the back propagation algorithm. This algorithm seeks to maximize the frame-level cross-entropy between the actual and projected probability distributions concerning class labels. Fig 3 shows the Architecture diagram for DBN.

The advent of RNNs (Recurrent Neural Networks), as referenced in [33], has brought about a transformative impact on the landscape of data processing. These networks introduce a potent instrument for managing sequential and time-dependent data. In contrast to conventional feed forward neural networks, RNNs exhibit a distinctive capability: they can effectively grasp context and interdependencies spanning diverse time intervals. The extracted feature \( R_s^{ef} \) is the input to this phase; RNNs contain an internal memory mechanism that allows them to maintain information about previous inputs in order to influence the
interpretation of current inputs. This memory mechanism forms the basis for their capability to model temporal relationships and patterns within data.

A RNN is a type of neural network comprising a concealed state denoted as \( k_p \) and an optionally produced output \( y \). This network functions on a sequence \( l \) of varying length, where \( l = [l_1, l_2, \ldots, l_p] \). In each sequential time step \( p \) the RNN’s hidden state, symbolized as \( k_p \), undergoes an update through the following Eq. (14):

\[
k_p = F(k_{p-1}l_p)R_s^y
\]  

(14)

The activation function \( F \) involved in this process can encompass a range of non-linearity. It might be as straightforward as a point wise logistic sigmoid function or as intricate as a LSTM unit.

An RNN possesses the capability to acquire knowledge of a probability distribution across a sequence through its training to predict the subsequent symbol within that sequence. In this scenario, the output at each time step \( p \) corresponds to the conditional distribution, denoted as \( m = [l_p, l_{p-1}, \ldots, l_1] \). To illustrate, a multinomial distribution employing a "1-of-K coding" technique can be generated using a softmax activation function shown in Eq.(15).

\[
m = [l_p, l_{p-1}, \ldots, l_1] = \frac{\sum_i u_i k_p}{\sum_i u_i k_p} R_s^y
\]  

(15)

Here, across all potential symbols \( j \) ranging from 1 to K, where \( u_i \) represents the rows of a weight matrix denoted as \( u \). Through the aggregation of these individual probabilities, we are enabled to calculate the overall probability of the sequence \( l \) using the following expression as in Eq.(16).

\[
m[l] = \prod_{p=1}^{P} m = [l_p, l_{p-1}, \ldots, l_1] R_s^y
\]  

(16)

From this acquired distribution, the process of generating a new sequence becomes straightforward. This is accomplished by iteratively selecting and sampling a symbol at each successive time step. While RNNs have brought significant advancements to data processing, they are not without challenges. The vanishing gradient problem, which hinders learning long-range dependencies, and the computational intensity of training RNNs on large datasets have driven the development of newer architectures, like Transformers. These models have become prominent in NLP tasks due to their parallel processing capabilities and attention mechanisms. Fig 4 shows the architecture diagram for RNN.
5.2 Recommended DB-RNN

The DB-RNN is a hybrid architecture that combines the strengths of DBNs and RNNs for time series classification tasks. Here the input is $R_f^S$ and the average predicted score obtained from the RNN and DBN network. This architecture aims to leverage the hierarchical feature learning capabilities of DBNs and the sequential modeling capabilities of RNNs to improve the accuracy and robustness of time series classification. Fig 5 shows the pictorial view of recommended DB-RNN for time series classification.
6 Results and discussion

6.1 Simulation Setup

In this work, Python was employed for the development of a time series classification framework, which was undergoing thorough analysis. Our approach integrated several traditional algorithms, namely Tuna Swarm Optimization (TSO)-DB-RNN [27], Beluga Whale Optimization (BWO)-DB-RNN [28], Cuttle Fish Optimization (CO)-DB-RNN [29], and Billiards-Inspired Optimization (BIO)-DB-RNN [26], into the proposed model. Additionally, we conducted a comprehensive comparison of classifiers, including RBF [30], CNN-DNN [31], CNN [5], DBN [32], RNN [33], and ICBPOA-ECDNN. For our experiments, set the population size at 10, established a maximum iteration limit of 50, and defined a chromosome length of 1. These parameters were carefully selected to drive the performance of our time series classification model.

6.2 Performance Measures

a) Sensitivity is in Eq. (17)

\[ sv = \frac{yyuu}{llkk + mmpp} \]  

(17)

(b) Specificity is in Eq. (18)

\[ sc = \frac{llkk}{llkk + yyuu} \]  

(18)

(c) FPR in Eq. (19)

\[ fr = \frac{jjii}{jjii + kkrr} \]  

(19)

(d) FNR is in Eq. (20)

\[ fr = \frac{mmpp}{mmpp + yyuu} \]  

(20)

(e) NPV is in Eq. (21)

\[ N = \frac{jjtt}{jjtt + llkk} \]  

(21)

(f) FDR is in Eq. (22)

\[ F = \frac{yyuu}{yyuu + kkii} \]  

(22)

(g) F1 score is in Eq. (23)

\[ \text{f1} = \frac{DK \cdot PL}{PL + DK} \]  

(23)

(h) Accuracy is in Eq. (23)
The “false negative and false positive” metrics are displayed as $llkk$ and $mmpp$ correspondingly. Also, the “true negative and true positive” metrics are displayed as $yyuu$ and $kkii$ appropriately.

6.3 Determining time series classification systems using a diverse algorithm

In Fig.6, the evaluation of the proposed time series classification system utilizing various algorithm models is presented. Furthermore, Fig 7 displays the results of the proposed time series classification system employing diverse classifier models. When examining the accuracy values represented graphically at a learning percentage of 50, the proposed RO-BIO-DB-RNN model outperforms better than traditional models such as TSO-DB-RNN, BWO-DB-RNN, CO-DB-RNN, and BIO-DB-RNN models by of 7\%, 5\%, 4\%, and 3\% respectively. This highlights the significant improvements achieved by the novel time series classification system model, affirning its superior performance.
Fig. 6. Determining the given time series classification system compared with classical algorithm models regarding “(a) Accuracy b) F1-Score, c) FDR, d) FNR, e) FPR, f) MCC, g) NPV, h) Precision i) Sensitivity and j) Specificity”
Fig. 7. Determining the given time series classification system compared with classical classifier models regarding “(a) Accuracy b) F1-Score, c) FDR, d) FNR, e) FPR, f) MCC, g) NPV, h) Precision i) Sensitivity and j) Specificity”

6.4 Actual and predicted analysis of proposed time series classification system

Fig. 8 delves the comparison between actual and predicted outcomes within the context of our proposed time series classification system. By leveraging this actual versus predicted analysis, effectively gauge the effectiveness of our novel approach in accurately classifying time series data and making reliable predictions. The alignment of actual and predicted results serves as a cornerstone for validating the credibility and applicability of our proposed time series classification system.
Fig. 8. Predicted and actual analysis of proposed time series classification system regarding (a) TSO-DB-RNN, (b) BWO-DB-RNN, (c) CO-DB-RNN, (d) BIO-DB-RNN and (e) BIO-DB-RNN.
6.5 Overall validation of the proposed time series classification system using diverse algorithms and classifiers

In Table 2, the determination of the proposed time series classification system model in accordance with diverse algorithm models is depicted. The suggested model's precision is 2%, 4%, 7% and 6% of TSO-DB-RNN, BWO-DB-RNN, CO-DB-RNN and BIO-DB-RNN respectively, which is lower than suggested RO-BIO-DB-RNN. Hence, a higher accuracy value ensures that the system will proficiently classify time series.

<table>
<thead>
<tr>
<th>Multiclass-Algorithm Comparison</th>
<th>Measures</th>
<th>TSO-DB-RNN</th>
<th>BWO-DB-RNN</th>
<th>CO-DB-RNN</th>
<th>BIO-DB-RNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Accuracy&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;Recall&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;Specificity&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;Precision&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;FPR&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;FNR&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;NPV&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;FDR&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;F1-Score&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;MCC&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Multiclass-Classifier Comparison</th>
<th>Measures</th>
<th>RBF</th>
<th>CNN</th>
<th>DBN</th>
<th>RNN</th>
<th>ICBPOA-ECDNN</th>
<th>RO-BIO-DB-RNN</th>
</tr>
</thead>
<tbody>
<tr>
<td>&quot;Accuracy&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;Recall&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;Specificity&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;Precision&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;FPR&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;FNR&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;NPV&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;FDR&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;F1-Score&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>&quot;MCC&quot;</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

7 Conclusion
accuracy was 2.8 %, 4.9 %, 7.2 % and 6.3 % of TSO-DB-RNN, BWO-DB-RNN, CO-DB-RNN and BIO-DB-RNN accordingly, which was lower than proposed RO-BIO-DB-RNN.

Thus, a greater accuracy number guarantees that the system was categorized time series effectively. Time series data often come with noise, missing values, and outliers. These can negatively impact classification accuracy and result in misclassifications. While time series classification has made significant progress, there are challenges to address and exciting avenues for future research and development. The integration of advanced techniques from deep learning, attention mechanisms, and other fields can lead to improved accuracy and applicability in various domains.

References


30. Meng Joo Er, Shiqian Wu, Juwei Lu, and Hock Lye Toh, Member " Face Recognition With Radial Basis Function (RBF) Neural Networks" NEURAL NETWORKS, VOL. 13, NO. 3, MAY 2002


32. Abdel-rahman Mohamed 1*, Dong Yu2 and Li Deng " Investigation of Full-Sequence Training of Deep Belief Networks for Speech Recognition" INTERSPEECH 2010
