Knowledge-Based Systems 260 (2023) 110158

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

MICOS: Mixed supervised contrastive learning for multivariate time series classification

Shilei Hao [a,b], Zhihai Wang [a,b], Afanasiev D. Alexander [c], Jidong Yuan [a,*], Wei Zhang [d]

[a] *School of Computer and Information Technology, Beijing Jiaotong University, 100044, Beijing, China*
[b] *Engineering Center on Computer Network Management in High Speed Railway, Ministry of Education, 100044, Beijing, China*
[c] *Laboratory of Artificial Intelligence and Machine Learning, Institute of Information Technology and Data Science, Irkutsk National Research Technical University, 664074, Irkutsk, Russia*
[d] *School of Artificial Intelligence, Guilin University of Electronic Technology, 541004, Guilin, China*

A R T I C L E I N F O

Article history:
Received 9 April 2022
Received in revised form 20 November 2022
Accepted 25 November 2022
Available online 28 November 2022

Keywords:
Multivariate time series
Classification
Representation learning
Mixed supervised contrastive learning

A B S T R A C T

Multivariate time series (MTS) classification is an emerging field with increasing demand. Existing representation learning methods for MTS classification are generally based on self-supervised learning. This results in their inability to maximize the use of labels. In addition, the inherent complexity of MTS makes it difficult to learn latent features. To this end, we introduce a new **M**ixed **S**upervised **C**ontrastive **L**oss (MSCL) for MTS representation learning. To effectively leverage labels, the MSCL is calculated by mixing self-supervised, intra-class and inter-class supervised contrastive learning approaches. Then, based on MSCL, we further propose a novel **MI**xed supervised **CO**ntrastive learning framework for MT**S** classification (MICOS). It uses the spatial and temporal channels to extract the complicated spatio-temporal features of MTS. Additionally, the MSCL is applied at the timestamp level to capture the multiscale contextual information. Experiments were carried out by performing supervised and self-supervised classification tasks on 30 public datasets from the UEA MTS archives. The results show the reliability and efficiency of MICOS compared with 13 competitive baselines.

1. Introduction

Multivariate time series (MTS) is an important type of data that is ubiquitous in a wide variety of domains, ranging from human activity recognition [1], health care [2], and energy management [3] to smart cities, financial markets [4], and other scientific and social domains. MTS classification is one of the most fundamental tasks of MTS. Unlike temporally invariant classification tasks, MTS classification is characterized by more complex settings due to the various temporal ordering correlations in MTS and their high dimension. All these issues lead to a considerable constraint for MTS classification in real-life scenarios.

Many MTS classification approaches have been proposed over the years. Distance-based methods, such as Dynamic Time Warping (DTW) with 1-Nearest Neighbor (1-NN) [5], and feature-based methods, such as the bag of Symbolic Fourier Approximation (SFA) [6], have been proven to be successful in MTS classification tasks. However, these approaches require heavy crafting in data preprocessing and feature engineering. More seriously, the predefined features usually fail to capture MTS latent characteristics. Recently, deep learning-based methods have been proposed to perform MTS classification with various network structures, such as TapNet [7] and ShapeNet [1], showing promising performance. Nonetheless, the above-mentioned networks have a rather limited ability to extract the spatio-temporal information in MTS. As shown in Fig. 1(a), samples of different classes show significant differences in spatial and temporal dimensions, indicating that the lack of spatio-temporal information may affect the discriminability of latent representations.

Recent research has turned to representation learning when handling MTS classification tasks, which allows the learning of class-separable embeddings in a self-supervised manner. It employs triplet losses [8] or contrastive loss [4] to optimize the models and generate the representation space; then, even a Support Vector Machines (SVM) [9] classifier is powerful enough on the learned representations. In particular, contrastive learning has recently shown its superiority for representation learning in time series classification [4,10] because of its ability to learn invariant representations. The common contrastive strategy in these works is as follows: pulling together each anchor and its "positive" samples, and pushing apart the anchor from its "negative" samples. This approach makes MTS more class-separable in the new representation space to a certain extent. However, as shown in Fig. 1(b), due to the lack of labels, both the anchor (A) and positive

* Corresponding author.
E-mail addresses: 18112020@bjtu.edu.cn (S. Hao), zhhwang@bjtu.edu.cn (Z. Wang), afanasiev-ad2009@yandex.ru (A.D. Alexander), yuanjd@bjtu.edu.cn (J. Yuan), wzhang@guet.edu.cn (W. Zhang).

https://doi.org/10.1016/j.knosys.2022.110158
0950-7051/© 2022 Elsevier B.V. All rights reserved.

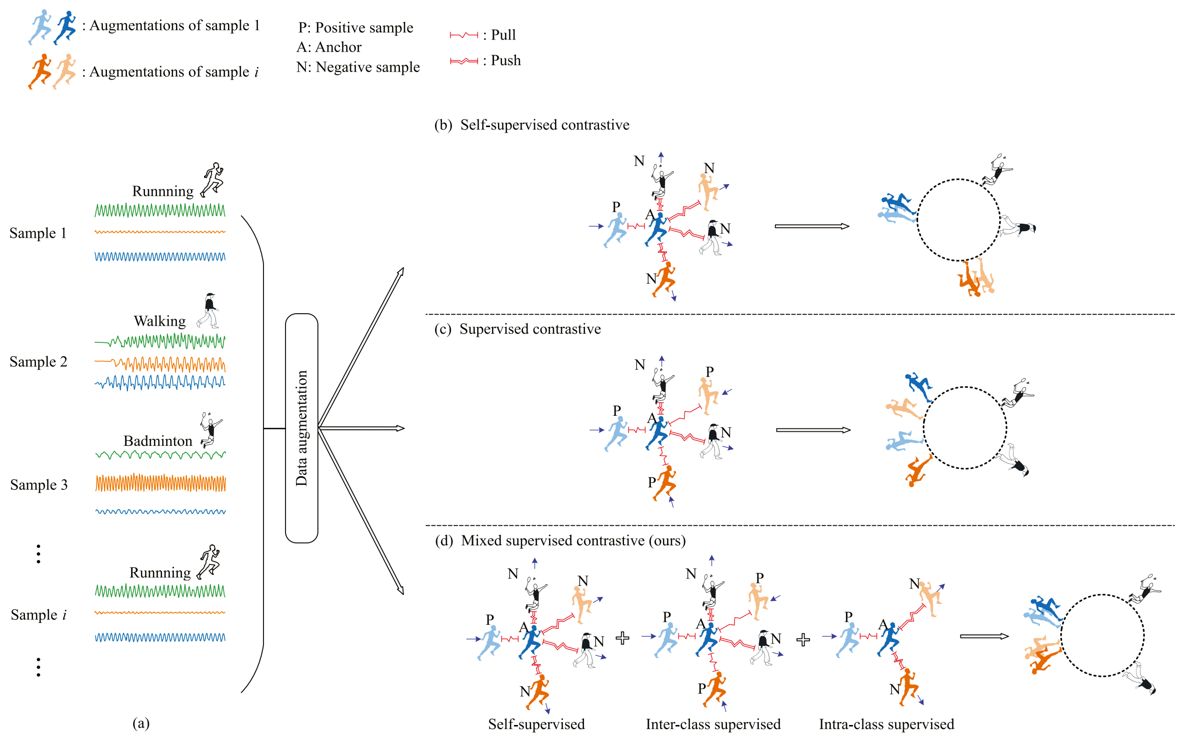

Fig. 1. Self-supervised contrastive vs. supervised contrastive vs. mixed supervised contrastive methods. The three different shapes of character graphics in the figure represent three different action classes: running, walking and badminton. Particularly, the blue and light blue augmented samples are from the original running sample, sample 1, and the orange and light orange samples are from the running sample *i*. The distance between two points on the hyperplane (ring) represents the similarity of the two samples.

(P) are augmentations of the same running sample, sample 1, while the negative (N) samples are all the samples (including augmentations of the running sample *i*) remaining in the batch. This situation may lead to the separation of instances from the same class in the new representation space and finally hinder the classification tasks. Given this, we argue that MTS representation learning has considerable room for improvement by utilizing the labels in the contrast process. In support, [11] proposed a supervised contrastive learning method for image classification, which contrasts the set of all augmented samples from the same class as positives against the negatives from the remainder of the batch. However, the fully supervised contrastive learning strategy (as shown in Fig. 1(c)) cannot pull augmentations from the same sample closer than those of the same class. Unlike [11], which only performs inter-class comparisons, we argue that it is necessary to consider intra-class contrastive learning to further pull homologous instances (augmentations from the same sample) together.

To address these issues, we propose a novel **MI**xed supervised **CO**ntrastive learning framework for MT**S** (MICOS), which learns representations by mixing self-supervised and supervised methods. To exploit the labels, MICOS applies a Mixed Supervised Contrastive Loss (MSCL) to optimize the network. As shown in Fig. 1(d), MSCL includes the self-supervised part and the supervised part. For the former, MSCL directly compares augmentations from the same original time series by taking advantage of the self-supervised approach without the use of labels. For the latter, the comparison is divided into two subsections: the inter-class subsection contrasts the set of all augmentations from the same class as positives against the negatives from the remainder of the batch; the intra-class subsection contrasts the set of all augmentations from the same original sample as positives against the negatives from the remainder of the class. As Fig. 1(d) shows, MSCL not only pulls together the augmentations from the same class but also brings the augmentations from the same original sample closer together, making it superior to the above two contrastive methods. In addition, we design a spatial channel and a temporal channel for MICOS to capture the spatio-temporal information of MTS. Finally, to capture multiscale contextual information, the MSCL is calculated hierarchically at the timestamp level. The main contributions of our work are summarized as follows:

- We propose the mixed supervised contrastive loss (MSCL), which exploits data labels to perform intra-class and inter-class comparison on samples to improve the performance of the traditional self-supervised contrastive learning method.
- We present MICOS, a novel framework to learn representations for MTS classification. The two-channel network guarantees the reliability of MICOS in extracting the spatial–temporal features. To the best of our knowledge, this is the first work that provides a reliable representation method by mixing self-supervised and supervised methods.
- Extensive experiments on 30 UEA benchmark datasets show the significant improvements of MICOS over state-of-the-art methods. Additional ablation studies demonstrate the benefits of utilizing labels in MTS representation learning.

The remainder of this paper is organized as follows. Section 2 reviews the most related work on MTS classification. Section 3 introduces the necessary concepts and the proposed MICOS method. Section 4 conducts a series of experiments to verify the proposed model, which is followed by the conclusion.

2. Related work

In this section, we briefly describe the recent advances in multivariate time series classification tasks.

2.1. Multivariate time series classification

Multivariate time series classification has been extensively studied in the literature [12]. In general, these methods can be grouped into three categories: distance-based, feature-based, and deep learning-based methods. Distance-based methods use 1-NN with various distance measures, such as the Euclidean Distance (ED), dimension-independent Dynamic Time Warping (DTW_I), and dimension-dependent Dynamic Time Warping (DTW_D), for classification [5,13,14]. For feature-based methods, [15] combined the shapelet representation from different variables to build an ensemble-like learner. HULM [16] uses binary stochastic hidden units to model latent structure in the data. A tree classifier based on symbolic representation was proposed in [17] to extract information contained in the relationships for MTS. An accurate and efficient classification method based on common principal components analysis was proposed in [18] to reduce the dimensionality for MTS. WEASEL+MUSE [6] uses the bag of SFA symbol model to classify MTS. Although these methods have been successfully applied to MTS classification, extensive data preprocessing and feature engineering are unavoidable, and their inherent nature makes it difficult to capture both the temporal dependency and the interactions of multiple dimensions in MTS.

Recently, deep learning-based methods [12] have achieved great success in MTS classification tasks. Zheng et al. [19] proposed multichannel deep convolutional neural networks (CNNs) that use 1D convolution to extract features from each dimension and then combined them with a Fully Connected (FC) layer. As an improvement, CA-SFCN [20] employs a 2D convolution with a cross attention mechanism to enhance the dependencies captured by a 1D convolution on spatial and temporal axes. In addition, some models adopt the Long Short-Term Memory (LSTM) [21] and the CNN together to learn representations for the time series. For example, the MLSTM-FCNs [22] employs an LSTM layer and a stacked CNN layer to extract features for classification. Similarly, TapNet [7] aggregates the LSTM layer, the stacked CNN layer, and the attentional prototype network to learn the latent features from MTS. Both of them enhanced the capture of the spatio-temporal information through a two-channel structure. However, their spatio-temporal interactions are computed at the instance level, which is different from our method at the timestamp level. Moreover, the random dimension permutation and grouping before encoding in TapNet may lead to the incomplete capture of spatial information. Additionally, ShapeNet [1] uses cluster-wise triplet loss to find discriminative shapelets. Although this method has considerable performance, it does not pay enough attention to the extraction of spatio-temporal information and is particularly time-consuming.

2.2. Multivariate time series representation learning

Representation learning based on contrastive learning approaches has achieved good performance in natural language processing [23], computer vision [11,24,25], and speech recognition [26]. In the time series domain, T-Loss [8] employs time-based negative sampling and a triplet loss to learn scalable representations for multivariate time series. More advantageously, TNC [27] leverages the local smoothness of a signal to define neighborhoods in time and learns generalizable representations of time series, showing better performance than T-Loss. TS-TCC [10] learns time series representations from unlabeled data via temporal and contextual contrasting. Similarly, TS2Vec [4] learns time series representations at the timestamp level via a hierarchical self-supervised contrastive loss. Additionally, the transformer-based TST [28] performs time series encoding with an unsupervised pre-training scheme and a masked mean-squared loss in the learning process. In general, existing approaches mostly adopt self-supervised methods to learn representations for time series classification tasks, which limits their ability to maximize the inter-class dissimilarity and ignores the intra-class diversity. Differently, we propose a representation learning framework that integrates self-supervised and supervised contrastive approaches. The use of labels ensures that the learned representations are more inter-class separable, and the supervised intra-class loss maintains their intra-class diversity. Previous works in computer vision also considered the intra-class diversity [24,25]. However, the contrastive approach in [24] is for finding less strongly correlated patterns by encouraging intra-class diversity, and its selection of negative samples depends on other classes. The intra-class comparison in [25] controls the comparison sample within the same image range and the contrastive loss is still carried out among pixels of multiple classes. In contrast, the supervised intra-class loss in our model is calculated in the same class, with the augmentations from the same sample as positives, and the remainder of the class as negatives.

3. Proposed method

In this section, we first formulate the definitions of MTS and representation learning for MTS classification. Then, the proposed framework MICOS is presented in detail.

3.1. Problem formulation

Definition 1 (*Multivariate Time Series*)**.** A multivariate time series (MTS) $X \in \mathcal{R}^{T\times M}$ is a sequence of real-valued vectors, $X = \{\boldsymbol{x}_1, \boldsymbol{x}_2, \dots, \boldsymbol{x}_T\}$, where $\boldsymbol{x}_t \in \mathcal{R}^M$, M is the feature dimension, t is the timestamp and T is the length of the time series.

Definition 2 (*Representation Learning for Multivariate Time Series Classification*)**.** Given a collection of multivariate time series $\mathcal{X} = \{X_1, X_2, \dots, X_N\}$ of N instances and its corresponding label set $\mathcal{Y} = \{y_1, y_2, \dots, y_N\}$, representation learning for classification aims to obtain category-specific representations $\mathcal{Z} = \{Z_1, Z_2, \dots, Z_N\}$ by learning a nonlinear embedding function $f_\theta : X_i \to Z_i$. The representation vector $Z_i = \{\boldsymbol{z}_{i,1}, \boldsymbol{z}_{i,2}, \dots, \boldsymbol{z}_{i,T}\}$ contains representation vectors $\boldsymbol{z}_{i,t} \in \mathcal{R}^K$ for each timestamp t, where K is the dimension of representation vectors and T is the length of the time series.

Once the model is trained, we can transform the raw MTS into a new representation space $\mathcal{Z}$ and use classifiers such as SVM for classification. Based on this principle, in this paper, the aim of our research is to learn class-separable representations for MTS classification tasks.

3.2. Model architecture

In this section, we present the proposed MICOS in detail. A schematic of MICOS is shown in Fig. 2.

3.2.1. Global structure of MICOS

As shown in Fig. 2, MICOS mainly includes the following steps. First, referring to the data augmentation method in [4], we randomly sample subseries with overlap from input MTS X_i as augmented samples. Second, these augmentations are fed into the encoder $f_\theta(\cdot)$, which consists of an embedding layer, a masking layer, a spatial channel, and a temporal channel. Then, the encoder embeds the input subseries into a new representation space by pooling and concatenating the spatial embedding and temporal embedding. Third, the MSCL is employed to optimize the model. Finally, we transform the original dataset into a new representation space and perform the classification task with an SVM classifier. To show the details, the classification process of MICOS is described in Algorithm 1.

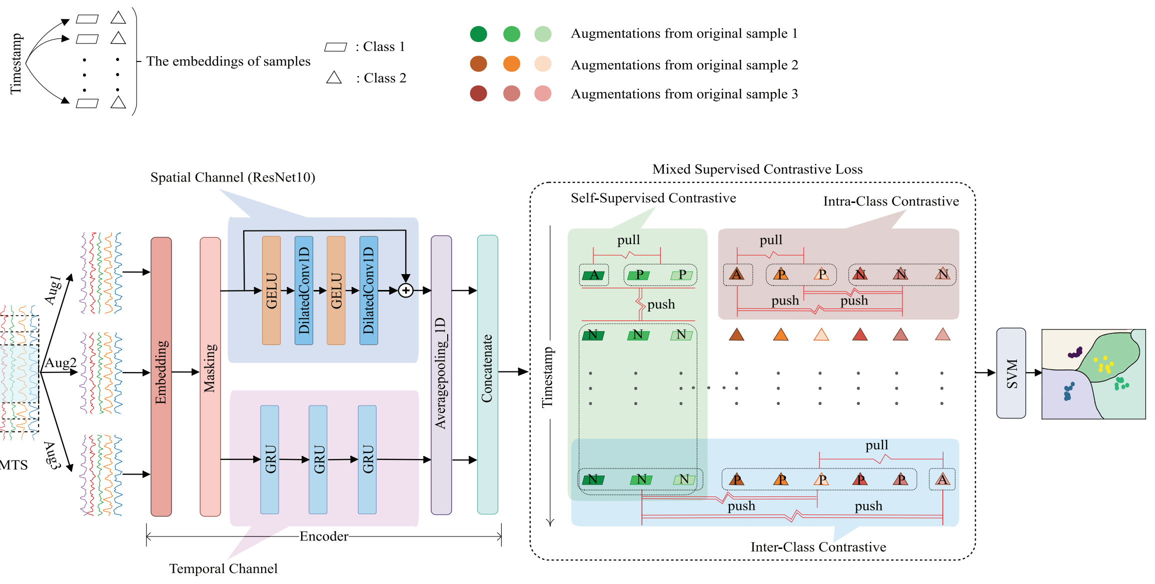
Timestamp
The embeddings of samples
: Class 1
: Class 2
Augmentations from original sample 1
Augmentations from original sample 2
Augmentations from original sample 3
Spatial Channel (ResNet10)
Mixed Supervised Contrastive Loss
Self-Supervised Contrastive
Intra-Class Contrastive
Inter-Class Contrastive
Aug1
Aug2
Aug3
MTS
Embedding
Masking
GELU
DilatedConv1D
GRU
Averagepooling_1D
Concatenate
Encoder
Temporal Channel
pull
push
SVM

Fig. 2. Model structure of MICOS. In the mixed supervised contrastive loss, the different colors indicate different augmented samples. In particular, the three shades of green represent augmentations from the same original sample. The same is true for three different shades of orange and red.

Algorithm 1 The classification process of MICOS

```
Input: training set: $\mathcal{X} \in \mathcal{R}^{T\times M\times N}$, the labels of training set: $\mathcal{Y} \in \mathcal{R}^{N}$, test set: $\hat{\mathcal{X}} \in \mathcal{R}^{T\times M\times \hat{N}}$, epochs: $E$, batch size: $B$, learning rate: $\eta$, the window size of average pooling: $l$, encoder structure $f_\theta(\cdot)$, the number of augmentations for each $X$: $A$.
Output: the labels of test set: $\hat{\mathcal{Y}} \in \mathcal{R}^{\hat{N}}$
1:  for all $i \in [0, E)$ do
2:      for sampled batch $\{X_j\}_{j=1}^{B}$ in $\mathcal{X}$ do
3:          $\{X_{j,1}, X_{j,2}, \ldots, X_{j,A}\} \leftarrow Augmentation(X_j), j \in [1, B]$
4:          for all $j \in \{1, 2, \ldots, B\}$ and $a \in \{1, 2, \ldots, A\}$ do
5:              $V_{j,a} \leftarrow Masking\left(Embedding\left(X_{j,a}\right)\right)$
6:              $S_{j,a} \leftarrow Spatio\_encoding(V_{j,a})$
7:              $H_{j,a} \leftarrow Temporal\_encoding(V_{j,a})$
8:              $S_{j,a} \leftarrow Averagepooling\_1D(S_{j,a})$
9:              $H_{j,a} \leftarrow Averagepooling\_1D(H_{j,a})$
10:             $Z_{j,a} \leftarrow concat\left(S_{j,a}, H_{j,a}\right)$
11:         end for
12:         use Eqs. (3)–(5) to calculate $\mathcal{L}_{self}^{(i,t)}$, $\mathcal{L}_{sup,intra}^{(i,t)}$, and $\mathcal{L}_{sup,inter}^{(i,t)}$
13:         recursively apply max pooling and Eq. (6) to calculate $\mathcal{L}_{MSCL}$
14:         update networks $f_\theta(\cdot)$ to minimize $\mathcal{L}_{MSCL}$
15:     end for
16: end for
17: $\mathcal{Z} \leftarrow f_\theta(\mathcal{X}), \hat{\mathcal{Z}} \leftarrow f_\theta(\hat{\mathcal{X}})$
18: build the SVM with $\mathcal{Z}$ and $\mathcal{Y}$
19: classify $\hat{\mathcal{Z}}$ with the SVM
20: return the labels $\hat{\mathcal{Y}}$
```

3.2.2. Spatio-temporal encoding on MTS

The encoding process of the encoder $f_\theta(\cdot)$ is shown in Lines 5–10 of Algorithm 1. Concretely, for each input X, the embedding layer *Embedding* $(\cdot)$, which is a fully connected layer, first maps the observation $\boldsymbol{x}_t$ at timestamp t to a latent vector $\boldsymbol{v}_t$. Then, the masking layer *Masking* $(\cdot)$ masks $V = \{\boldsymbol{v}_1, \boldsymbol{v}_2, \ldots, \boldsymbol{v}_T\}$ with a binary mask that is randomly generated from a Bernoulli distribution with $p = 0.5$ along the time axis. After *Masking* $(\cdot)$, $\boldsymbol{v}_t$ is embedded by the spatial and temporal channels. Notably, the fully connected layer and masking layer are adopted to generate an augmented context view. For the interested reader, a detailed analysis can be found in [4].

As shown in Fig. 2, for the spatial channel *Spatio_encoding* $(\cdot)$, residual networks containing 10 hidden blocks and an output block are used to extract the spatial features. The residual block can be formulated as

$$\boldsymbol{v}_t^{i+1} = \mathcal{F}\left(\boldsymbol{v}_t^{i}, \{W_i\}\right) W_s \boldsymbol{v}_t^{i}, \tag{1}$$

where W_i and W_s are weight parameters, i is the order of residual blocks, and $\mathcal{F}(\cdot)$ is used to compute the residuals. Specifically, every residual block consists of two 1-D dilated convolutional layers with a Gaussian Error Linear Unit (GELU) [29] activation function, and there are skip connections between adjacent blocks. For the k-th block, the dilation rate of the convolution is set to 2^k, and the kernel size is fixed to 3. Then, the average pooling layer *Averagepooling_1D* $(\cdot)$ is applied to the output of the residual module, and the input V is mapped to $S \in R^{T\times d_s}$, where d_s is the output dimension of the output block.

For temporal channel *Temporal_encoding* $(\cdot)$, among different variants of recurrent neural networks, we specifically consider Gated Recurrent Units (GRUs), which are relatively simple, have few parameters, and are less prone to overfitting. In the GRUs, an update gate ($\boldsymbol{u}_t$) and a reset gate ($\boldsymbol{r}_t$) control the hidden state $\boldsymbol{h} \in R^{d_t}{}_t$ with $\boldsymbol{v}_t$. Similar to $\boldsymbol{h}_t$, the dimension of $\boldsymbol{h}_{t-1}$ can be expressed as $\boldsymbol{h} \in R^{d_t}{}_{t-1}$, where d_t is the output dimension of the GRUs. The update functions are as follows:

$$\begin{aligned}
\boldsymbol{r}_t &= \sigma\left(W_{v,r}\boldsymbol{v}_t + \boldsymbol{b}_{v,r} + W_{h,r}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{h,r}\right),\\
\boldsymbol{u}_t &= \sigma\left(W_{v,u}\boldsymbol{v}_t + \boldsymbol{b}_{v,u} + W_{h,u}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{h,u}\right),\\
\boldsymbol{n}_t &= \tanh\left(W_{v,n}\boldsymbol{v}_t + \boldsymbol{b}_{v,n} + \boldsymbol{r}_t \circ \left(W_{h,n}\boldsymbol{h}_{t-1} + \boldsymbol{b}_{h,n}\right)\right),\\
\boldsymbol{h}_t &= (1 - \boldsymbol{u}_t) \circ \boldsymbol{n}_t + \boldsymbol{u}_t \circ \boldsymbol{h}_{t-1},
\end{aligned} \tag{2}$$

where $W_{\{v,r|v,u|vn|h,r|h,u|h,n\}}$ is the model parameter, σ is the sigmoid function, $\circ$ is the element-wise multiplication operation and $\boldsymbol{b}_{\{v,r|v,u|vn|h,r|h,u|h,n\}}$ is the bias term. Three GRUs are cascaded with the average pooling layer to output $H \in R^{T\times d_t}$. Then, we concatenate the results of the spatial channel and temporal channel together and obtain the combined spatio-temporal features embedding $Z = concat\,(S, H) \in R^{T\times(d_t+d_s)}$. The obtained new embedding contains both the temporal and spatial information in the MTS.

3.2.3. Mixed supervised contrastive loss

To maximize the class separability of $\mathcal{Z}$, the MSCL contains one self-supervised contrastive loss and two supervised contrastive losses. To learn discriminative representations over time, as shown in Fig. 2, all these losses are calculated at the timestamp level. The details are as follows.

Self-Supervised Contrastive Loss: For the augmentations generated by the same original MTS, MICOS takes embeddings at the same timestamp as positive samples, and those at different timestamps as negative samples. Let i be the index of the input series X_i and t be the timestamp. Then, $\boldsymbol{z}_{i,t}$ and $\boldsymbol{z}_{a,t}$, $(a \neq i)$ denote the embedding of the same timestamp t from different augmentations of X_i. The self-supervised contrastive loss for the i-th MTS at timestamp t can be formulated as

$$\mathcal{L}_{self}^{(i,t)} = \frac{-1}{|A^*(i)|} \sum_{a\in A^*(i)} \log \frac{\exp\left(\boldsymbol{z}_{i,t}\cdot \boldsymbol{z}_{a,t}\right)}{\sum_{t'\in T_s}\left(\sum_{a\in A^*(i)} \exp\left(\boldsymbol{z}_{i,t}\cdot\boldsymbol{z}_{a,t'}\right) + \mathrm{I}_{[t\neq t']}\exp\left(\boldsymbol{z}_{i,t}\cdot\boldsymbol{z}_{i,t'}\right)\right)}, \tag{3}$$

where $A^*(i)$ is the set of indices of augmentations generated by X_i excluding $\boldsymbol{z}_i$, and $|A^*(i)|$ is its cardinality. T_s is the set of timestamps in the augmentation, and I is the indicator function.

Supervised Contrastive Loss: For supervised contrast, we claim that there are two key issues that need to be considered. First, all samples of the same class should be pulled together, and samples of different classes should be pushed apart. Second, for all samples of the same class in the batch, due to differences between samples, the augmentations from the same sample should be pulled together, and the augmentations from different samples should be pushed apart. Our views are visualized in Fig. 1(d). To address these two issues, as shown in Fig. 2, we propose the intra-class contrastive loss and inter-class contrastive loss, which are calculated at the timestamp level. Specifically, when computing the intra-class contrastive loss, we take the embedding at the same timestamp from the augmentations of the same input MTS as positive samples and those at the same timestamp from the augmentations of different input MTS (but belonging to the same class) as negative samples. The intra-class contrastive loss indexed with (i, t) can be formulated as

$$\mathcal{L}_{sup,intra}^{(i,t)} = \frac{-1}{|A^*(i)|} \sum_{a\in A^*(i)} \log \frac{\exp\left(\boldsymbol{z}_{i,t}\cdot\boldsymbol{z}_{a,t}\right)}{\sum_{j\in\bar{B}(i)}\sum_{q\in A(j)} \exp\left(\boldsymbol{z}_{i,t}\cdot\boldsymbol{z}_{q,t}\right) + \sum_{a\in A^*(i)}\exp\left(\boldsymbol{z}_{i,t}\cdot\boldsymbol{z}_{a,t}\right)}. \tag{4}$$

Here, $\bar{B}(i)$ is the set of indices of all input samples that belong to the same class as X_i in the batch excluding X_i. $A(j)$ is the set of indices of augmentations generated by X_j.

Unlike the intra-class contrastive loss, we treat the embedding at the same timestamp from the augmentations of the same class in the augmented batch as positive samples and other classes in the same batch as negative samples. Then, the contrastive loss indexed with (i, t) can be formulated as

$$\mathcal{L}_{sup,inter}^{(i,t)} = \frac{-1}{|\bar{P}(i)|} \sum_{p\in\bar{P}(i)} \log \frac{\exp\left(\boldsymbol{z}_{i,t}\cdot\boldsymbol{z}_{p,t}\right)}{\sum_{j\in\bar{B}}\sum_{q\in A(j)} \exp\left(\boldsymbol{z}_{i,t}\cdot\boldsymbol{z}_{q,t}\right) + \sum_{a\in A^*(i)}\exp\left(\boldsymbol{z}_{i,t}\cdot\boldsymbol{z}_{a,t}\right)}, \tag{5}$$

where $\bar{P}(i)$ is the set of indices of augmentations that belong to the same class as X_i in the augmented batch excluding $\boldsymbol{z}_i$ and $\bar{B}$ is the set of indices of all input samples excluding X_i.

All three losses are complementary to each other and jointly optimize the model for the extraction of time series features. The overall loss is defined as

$$\mathcal{L}_{MSCL} = \sum_i \sum_t \left(\mathcal{L}_{self}^{(i,t)} + \mathcal{L}_{sup,intra}^{(i,t)} + \mathcal{L}_{sup,inter}^{(i,t)}\right). \tag{6}$$

The calculation of MSCL is shown in Lines 12–13 of Algorithm 1. Note that time series are often recorded with different granularities, and a fixed window size may not work well for different types of datasets. Therefore, to learn the multiscale contextual information in MTS, we adopt the multiscale method of [4] to force the model to learn sequence embeddings at different scales. Concretely, max pooling is applied to the learned embeddings along the time axis and to compute Eq. (6), recursively. Finally, when the epochs are completed, MICOS maps all samples in the dataset into a new representation space, where the SVM performs the classification task (see Lines 17–19 of Algorithm 1 for an illustration).

3.3. Computational complexity

In this section, we discuss the computational complexity of the proposed loss function MSCL. The network structure of MICOS is similar to or even simpler than other methods, such as MLSTM-FCN [22] and TapNet [7], and their main difference is the computation of the loss function. Given a time series dataset with N samples of length T, the time complexity of the proposed MSCL can be roughly expressed as $O\left(NT^2 + N^2T\right)$. Although the MSCL is more complex than the cross-entropy loss adopted by TapNet and MLSTM-FCNs, it is normally as complex as T-Loss [8] and TS2Vec [4]. Moreover, ShapeNet [1], which also employs a contrastive approach, is more time-consuming than our model due to the additional shapelets searching process. In addition, the experimental results in Section 4 demonstrate that our model has the best performance in terms of accuracy.

4. Experiments

In this section, we conduct extensive experiments on 30 benchmark datasets for MTS classification and compare the results of our model, MICOS,[1] with those of the other 13 baselines.

Datasets The UEA MTS classification archive[2] consists of real-world multivariate time series data collected from different applications, such as human activity recognition, motion classification, and ECG/EEG (electroencephalogram) signal classification The dimension of the tested MTS ranges from two dimensions in trajectory classification data to 963 dimensions in the traffic flow classification task. The length of these time series ranges from 8 to 3000. The dataset sizes of these time series range from 27 to 10992. The detailed statistics of each dataset are summarized in Table 1.

Metrics For each dataset, we compute the classification accuracy as the evaluation metric. We also compute the average accuracy and the number of Wins/Draws/Losses to compare the different methods.

Methods for Comparison We compare MICOS with the following MTS classifiers. Interested readers may refer to the original paper for further information:

(1) **ED, DTW$_I$**, and **DTW$_D$** [5]: The three benchmark classifiers are based on the Euclidean distance, dimension-independent dynamic time warping and dimension-dependent dynamic time warping, respectively.

[1] The source code is publicly available at https://github.com/haoshileii/micos.

[2] The datasets can be found at www.timeseriesclassification.com.

Table 1
Statistics of the 30 UEA datasets used in experimentation.

Dataset	Train Cases	Test Cases	Dimensions	Length	Classes
ArticularyWordRecognition	275	300	9	144	25
AtrialFibrillation	15	15	2	640	3
BasicMotions	40	40	6	100	4
CharacterTrajectories	1422	1436	3	182	20
Cricket	108	72	6	1197	12
DuckDuckGeese	60	40	1345	270	5
EigenWorms	128	131	6	17984	5
Epilepsy	137	138	3	206	4
EthanolConcentration	261	263	3	1751	4
ERing	30	30	4	65	6
FaceDetection	5890	3524	144	62	2
FingerMovements	316	100	28	50	2
HandMovementDirection	320	147	10	400	4
Handwriting	150	850	3	152	26
Heartbeat	204	205	61	405	2
JapaneseVowels	270	370	12	29	9
Libras	180	180	2	45	15
LSST	2459	2466	6	36	14
InsectWingbeat	30000	20000	200	78	10
MotorImagery	278	100	64	3000	2
NATOPS	180	180	24	51	6
PenDigits	7494	3498	2	8	10
PEMS-SF	267	173	963	144	7
Phoneme	3315	3353	11	217	39
RacketSports	151	152	6	30	4
SelfRegulationSCP1	268	293	6	896	2
SelfRegulationSCP2	200	180	7	1152	2
SpokenArabicDigits	6599	2199	13	93	10
StandWalkJump	12	15	4	2500	3
UWaveGestureLibrary	120	320	3	315	8

(2) **WEASEL+MUSE** [6]: Word extraction for time series classification (WEASEL) with a Multivariate Unsupervised Symbols and dErivatives (MUSE) is the most effective bag-of-patterns algorithm for MTS classification.

(3) **MLSTM-FCN** [22]: MLSTM-FCNs is a deep learning framework that transforms the LSTM-FCN models of univariate time series into MTS by augmenting with a squeeze-and-excitation block.

(4) **TapNet** [7]: TapNet draws on the strengths of both traditional and deep learning approaches.

(5) **TS2Vec (Supervised)** [4,11]: TS2Vec (Supervised) applies the supervised contrastive learning method proposed in [11] to the original TS2Vec [4]. Specifically, we change the instance-wise contrastive loss in TS2Vec to a supervised version. In the calculation of the loss, for any anchor, we consider all samples of the same class as positive samples and different classes as negative samples. Meanwhile, the temporal contrastive loss is no longer adopted.

(6) **ShapeNet** [1]: ShapeNet is a novel shapelet-neural network approach for MTS classification, that is highly competitive in accuracy.

(7) **Self-supervised baselines:** To further explore the performance of our supervised methods, we compare MICOS with other existing unsupervised methods, including T-loss [8], TNC [27], TS-TCC [10], TST [28], and TS2Vec [4].

Training Details The proposed model is implemented with PyTorch 1.7.0 in Python 3.8.0 and trained with a TITAN Xp 12G GPU with CUDA 11.0. The settings of the main parameters are summarized in Table 2. For each input MTS, we generate three augmentations, i.e., A is set to 3. We train an SVM classifier with a radial basis function kernel on the learned embedding space. For the spatial channel, the size of the convolutional kernels in ResNet is set to 3, and the output dimension of the ResNet layer d_s is 320. For the temporal channel, we use a 3-layer GRU network, and the output dimension d_t is 128. For the pooling layer after the two channels, we adopt a 1-D average pooling layer, and the window size l is set to 7. The batch size B is set to 8, and the number of optimization epochs E is set to 500. In particular, we use AdamW optimization to optimize the parameters of our model, and the learning rate η is kept fixed at 0.0005.

Table 2
Parameter settings.

Parameters	Values	Description
d_s	320	The output dimension of the ResNet
d_t	128	The output dimension of GRUs
l	7	The window size of 1-D average pooling layer
B	8	The batch size
E	500	The number of optimization epochs
η	0.0005	The learning rate

4.1. Convergence of MICOS

Depending on the parameters from Section 4, we verify the convergence of MICOS. Concretely, we test the convergence on 4 UEA datasets, RacketSports, JapaneseVowels, ArticularyWordRecognition, and PEMS-SF, as illustrated in Fig. 3. It is clear that all the losses converge smoothly as training proceeds on all 4 datasets. The losses converge quickly at the beginning and then become stabilize. Similar trends can also be observed from other datasets. This verifies the effectiveness of our mixed hierarchical contrastive loss.

4.2. Hyperparameter stability

We experimented with hyperparameter stability by changing the batch size B, learning rate η, number of epochs E, and window size of pooling l one at a time from the best combination for each of the methodologies. Due to the large number of datasets, all experiments were performed by cross-validation on the training sets of 10 randomly selected datasets: RacketSports, Libras, NATOPS, ERing, Handwriting, HandMovementDirection, Heartbeat, SelfRegulationSCP2, Cricket, and StandWalkJump. In Fig. 4,

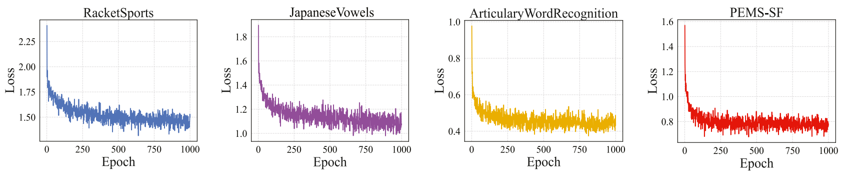

Fig. 3. Convergence of the learning algorithm on some MTS datasets.

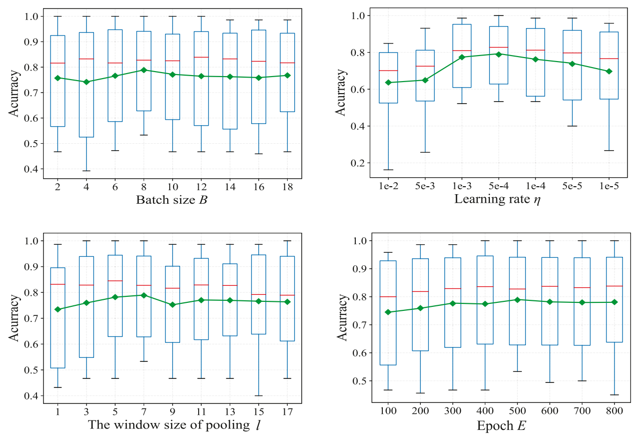

Fig. 4. Boxplot showing the accuracies on 10 UEA datasets vs. changes in the batch size B, learning rate η, epoch E and window size l of pooling.

the standard boxplots show the classification accuracies of all 10 datasets with respect to changes in B:{2, 4, 6, …, 18}, η:{0.01, 0.005, 0.001, …, 0.00001}, E:{100, 200, 300, …, 800} and l:{1, 3, 5, …, 17}. In particular, the purple curve in the figures reflects the change in the average accuracy on the 10 datasets. Fig. 4 shows that the performance of the model grows first and then drops to varying degrees with the changes in B, η, and l. This is because a batch that is too large or too small is not conducive to the convergence of the training process. The same is true for the learning rate. For the pooling window size, a small size is not conducive to extracting context information, and an excessively large window size may be ineffective for a shorter MTS. Finally, for the number of epochs, the performance of the model tends to stabilize after E=400. In general, to achieve the best performance of the model, we fixed the model hyperparameter B to 8, η to 0.0005, E to 500, and l to 7.

4.3. Classification performance evaluation

Tables 3 and 4 show the classification results on the 30 UEA datasets of MICOS and other supervised and self-supervised methods. The experimental accuracies of the baseline results all refer to the original papers or [4,12]. The results marked "N/A" in the table indicate that the corresponding approach is incapable of running the results due to memory or computational issues. The highest accuracy score in each dataset is bolded.

Comparison with supervised methods: The results in Table 3 show that for MTS classification problems, the deep learning methods, such as TapNet, MLSTM-FCN, and ShapeNet, proposed in recent years generally surpass the traditional methods (e.g., ED and DTW$_I$). In terms of the best accuracy, MICOS ranks first in 16 out of 30 datasets. For the comparisons of the 1-to −1-Wins/Draws/Losses, MICOS performs much better than all the baselines. The average accuracy of the best baseline, ShapeNet, is 0.714, which is approximately 4% lower than the average

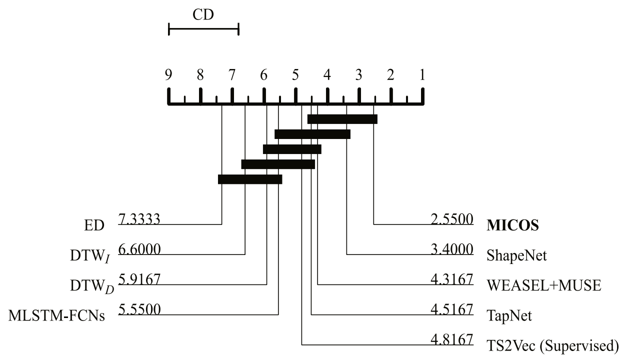

Fig. 5. Critical difference plot of the supervised MTS classifiers on the 30 UEA datasets with $\alpha = 0.05$.

accuracy of 0.742 of MICOS. In particular, compared with those of the current best baseline, the accuracies of MICOS on the LSST and PEMS-SF datasets are improved by more than 7%, and the improvement over HandMovementDirection is more than 70%. The reasons our MICOS outperforms other baselines may lie in the fact that the shapelets extraction network of ShapeNet restricts its ability of capturing spatio-temporal information, and the random dimension selection strategy of TapNet in the coding process reduces its information extraction ability. In addition, the proposed MSCL loss function is also crucial to improve the classification performance, which is illustrated by the ablation experiment in Section 4.4.

Following the process described in [30,31], we also conducted the Friedman test to evaluate the performances of all the methods. Fig. 5 shows the critical difference plot with α = 0.05 from the results shown in Table 3. The values correspond to the average rank, and the methods linked by a bar do not have a statistically significant difference. MICOS achieves the best overall average rank of 2.5500, which is lower than the ranks of the existing state-of-the-art approaches ShapeNet, WEASEL+MUSE, and TapNet. In addition, MICOS is significantly better than ED, DTW$_I$, DTW$_D$, MLSTM-FCNs, and TS2Vec (Supervised).

Comparison with self-supervised methods: We also compared MICOS with the existing novel self-supervised models TST, TNC, TS-TCC, T-Loss, and TS2Vec. The results in Table 4 show that our method performs better than all self-supervised baselines in terms of average accuracy, total best accuracy, and 1-to −1-Wins. Specifically, our model outperforms the current best self-supervised model, TS2Vec, on 21 datasets and outperforms it by an average accuracy of 5.4%. In addition, for a more comprehensive comparison, in addition to the original SVM classifier used to compare the learned representations, we also use a deep neural network (DNN) classifier to compare the two models with the same parameters. Referring to the settings in [7], we use two fully connected layers and the softmax function to classify the learned representation, and the results are shown in Table 4 as TS2Vec(DNN) and MICOS(DNN). The results show that the performances of both MICOS and TS2Vec decrease when using the DNN classifier, which can be attributed to the imperfect network

Table 3
Comparison of our proposed MICOS and baselines on UEA datasets.

Dataset	ED	DTW$_I$	DTW$_D$	MLSTM -FCNs	WEASEL +MUSE	TapNet	TS2Vec (Supervised)	ShapeNet	MICOS
ArticularyWordRecognition	0.970	0.980	0.987	0.973	**0.990**	0.987	0.980	0.987	**0.990**
AtrialFibrillation	0.267	0.267	0.220	0.267	0.333	0.333	0.200	**0.400**	0.333
BasicMotions	0.676	**1.000**	0.975	0.950	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CharacterTrajectories	0.964	0.969	0.989	0.985	0.990	**0.997**	0.992	0.980	0.994
Cricket	0.944	0.986	**1.000**	0.917	**1.000**	0.958	0.958	0.986	**1.000**
DuckDuckGeese	0.275	0.550	0.600	0.675	0.575	0.575	0.660	0.725	**0.740**
EigenWorms	0.549	N/A	0.618	0.504	0.890	0.489	0.840	0.878	**0.901**
Epilepsy	0.666	0.978	0.964	0.761	**1.000**	0.971	0.942	0.987	0.971
ERing	0.930	0.914	0.929	0.941	**0.964**	0.895	0.896	0.574	0.941
EthanolConcentration	0.293	0.304	0.323	0.373	**0.430**	0.323	0.285	0.323	0.247
FaceDetection	0.519	N/A	0.529	0.545	0.545	0.556	0.517	**0.602**	0.523
FingerMovements	0.550	0.520	0.530	**0.580**	0.490	0.530	0.540	**0.580**	0.570
HandMovementDirection	0.278	0.306	0.231	0.365	0.365	0.378	0.297	0.338	**0.649**
Handwriting	0.200	0.316	0.286	0.286	0.605	0.357	0.579	0.451	**0.621**
Heartbeat	0.619	0.658	0.717	0.663	0.727	0.751	0.737	0.756	**0.766**
InsectWingbeat	0.128	N/A	N/A	0.167	N/A	0.208	0.179	**0.250**	0.218
JapaneseVowels	0.924	0.959	0.949	0.976	0.973	0.965	0.976	0.984	**0.989**
Libras	0.833	**0.894**	0.870	0.856	0.878	0.850	0.867	0.856	0.889
LSST	0.456	0.575	0.551	0.373	0.590	0.568	0.545	0.590	**0.667**
MotorImagery	0.510	N/A	0.500	0.510	0.500	0.590	0.460	**0.610**	0.500
NATOPS	0.850	0.850	0.883	0.889	0.870	0.939	0.922	0.883	**0.967**
PEMS-SF	0.705	0.734	0.711	0.699	N/A	0.751	0.775	0.751	**0.809**
PenDigits	0.973	0.939	0.977	0.978	0.948	0.980	**0.981**	0.977	**0.981**
Phoneme	0.104	0.151	0.151	0.110	0.190	0.175	0.231	**0.298**	0.276
RacketSports	0.868	0.842	0.803	0.803	0.934	0.868	0.901	0.882	**0.941**
SelfRegulationSCP1	0.771	0.765	0.775	**0.874**	0.710	0.652	0.741	0.782	0.799
SelfRegulationSCP2	0.483	0.533	0.539	0.472	0.460	0.550	0.556	**0.578**	**0.578**
SpokenArabicDigits	0.967	0.959	0.963	**0.990**	0.982	0.983	0.978	0.975	0.981
StandWalkJump	0.200	0.333	0.200	0.067	0.333	0.400	**0.533**	**0.533**	**0.533**
UWaveGestureLibrary	0.881	0.868	0.903	0.891	**0.916**	0.894	0.913	0.906	0.891
Average accuracy	0.612	0.605	0.656	0.648	0.673	0.682	0.699	0.714	0.742
Total best accuracy	0	2	1	3	7	2	3	9	16
Ours 1-to-1-Wins	28	26	25	22	19	21	25	18	–
Ours 1-to-1-Draws	0	1	2	2	5	3	3	3	–
Ours 1-to-1-Losses	2	3	3	6	6	6	2	9	–

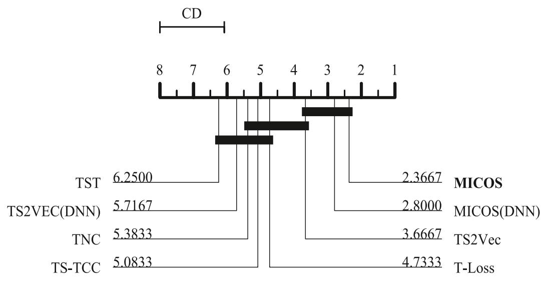

Fig. 6. Critical difference plot of the self-supervised MTS classifiers on the 30 UEA datasets with $\alpha = 0.05$.

structure of the classifier or the need for the better adjustment of parameters. However, even with the degraded performance, MICOS(DNN) still outperforms TS2Vec(DNN), which is sufficient to demonstrate the better performance of the proposed method.

Furthermore, Fig. 6 shows the critical difference plot from the results shown in Table 4. Although there is no significant difference between MICOS and TS2Vec, MICOS has a better average rank. MICOS(DNN) is significantly superior to TS2Vec(DNN). These two comparisons fully illustrate that the proposed MICOS has better representation learning performance than TS2Vec. In addition, it is obvious that MICOS outperforms TST, TNC, TS-TCC, and T-Loss.

4.4. Ablation experiment

To verify the effectiveness of the proposed components of MICOS, a comparison between MICOS and its 7 variants on 30 UEA datasets is shown in Table 5. Specifically, in addition to the full MICOS method, we derived different model variants for comparison as follows. We trained the model without the spatial channel, the temporal channel, the self-supervised contrastive loss, the intra-class contrastive loss, and the inter-class contrastive loss, denoted MICOS$_{out\_SC}$, MICOS$_{out\_TC}$, MICOS$_{out\_self}$, MICOS$_{out\_Intra}$, and MICOS$_{out\_Inter}$, respectively. In addition, we ran MICOS with only the self-supervised loss, which is called MICOS$_{self}$. Furthermore, to further demonstrate the effect of the proposed MSCL on improving the classification performance, we also replaced the MSCL in the MICOS model with the cross-entropy loss, which we call MICOS$_{Cross\_Entropy}$. Fig. 7 shows the critical difference plot of all the above variants. Obviously, MICOS achieves the best ranking compared with all of its variants, which demonstrates the effectiveness of the proposed components. In particular, MICOS$_{out\_Inter}$ ranks the worst among these variants. This illustrates the necessity of data labels in the computation of contrastive loss. The ranking of MICOS$_{self}$ is significantly lower than that of MICOS, which indicates that the supervised part of our proposed loss function is effective. Additionally, the performance of MICOS$_{Cross\_Entropy}$ being significantly worse than that of MICOS demonstrates the effect of the proposed MSCL. In summary, the above results further show the effect of the new technical points in the proposed MICOS.

4.5. Inspection of class prototype

In this section, we visualize the class prototype and its corresponding time series embedding to show the effectiveness of our well-trained time series embedding. First, we analyze the learned representations over time on the Epilepsy dataset from the UEA archive with the heatmap. The data were collected from

Table 4
Comparison of our proposed MICOS and baselines (self-supervised) on UEA datasets.

Dataset	TST	TS-TCC	TNC	T-Loss	TS2Vec	TS2Vec (DNN)	MICOS (DNN)	MICOS
ArticularyWordRecognition	0.977	0.953	0.973	0.943	0.987	0.980	**0.993**	0.990
AtrialFibrillation	0.067	0.267	0.133	0.133	0.200	0.125	**0.375**	0.333
BasicMotions	0.975	**1.000**	0.975	**1.000**	0.975	0.975	**1.000**	**1.000**
CharacterTrajectories	0.975	0.985	0.967	0.993	**0.995**	0.948	0.994	0.994
Cricket	**1.000**	0.917	0.958	0.972	0.972	0.972	0.986	**1.000**
DuckDuckGeese	0.620	0.380	0.460	0.650	0.680	0.625	**0.750**	0.740
EigenWorms	0.748	0.779	0.840	0.840	0.847	0.406	**0.945**	0.901
Epilepsy	0.949	0.957	0.957	0.971	0.964	0.949	**0.985**	0.971
ERing	0.874	0.904	0.852	0.133	0.874	0.886	0.939	**0.941**
EthanolConcentration	0.262	0.285	0.297	0.205	**0.308**	0.273	0.277	0.247
FaceDetection	0.534	**0.544**	0.536	0.513	0.501	0.505	0.543	0.523
FingerMovements	0.560	0.460	0.470	**0.580**	0.480	0.542	0.552	0.570
HandMovementDirection	0.243	0.243	0.324	0.351	0.338	0.264	0.639	**0.649**
Handwriting	0.225	0.498	0.249	0.451	0.515	0.448	0.617	**0.621**
Heartbeat	0.746	0.751	0.746	0.741	0.683	0.695	**0.790**	0.766
InsectWingbeat	0.105	0.264	**0.469**	0.156	0.466	0.192	0.222	0.218
JapaneseVowels	0.978	0.930	0.978	**0.989**	0.984	0.981	0.986	**0.989**
Libras	0.656	0.822	0.817	0.883	0.867	0.858	0.881	**0.889**
LSST	0.408	0.474	0.595	0.509	0.537	0.500	0.622	**0.667**
MotorImagery	0.500	**0.610**	0.500	0.580	0.510	0.563	0.521	0.500
NATOPS	0.850	0.822	0.911	0.917	0.928	0.926	0.955	**0.967**
PEMS-SF	0.740	0.734	0.699	0.676	0.682	0.601	0.792	**0.809**
PenDigits	0.560	0.974	0.979	0.981	**0.989**	0.987	0.980	0.981
PhonemeSpectra	0.085	0.252	0.207	0.222	0.233	0.203	0.248	**0.276**
RacketSports	0.809	0.816	0.776	0.855	0.855	0.763	0.921	**0.941**
SelfRegulationSCP1	0.754	0.823	0.799	**0.843**	0.812	0.719	0.740	0.799
SelfRegulationSCP2	0.550	0.533	0.550	0.539	**0.578**	0.540	0.403	**0.578**
SpokenArabicDigits	0.923	0.970	0.934	0.905	**0.988**	0.970	0.981	0.981
StandWalkJump	0.267	0.333	0.400	0.333	0.467	0.375	0.375	**0.533**
UWaveGestureLibrary	0.575	0.753	0.759	0.875	**0.906**	0.891	0.903	0.891
Average accuracy	0.617	0.668	0.670	0.658	0.704	0.655	0.731	0.742
Total best accuracy	1	3	1	4	6	0	7	14
Ours 1-to-1-Wins	26	24	25	23	21	27	16	–
Ours 1-to-1-Draws	2	1	2	4	1	0	3	–
Ours 1-to-1-Losses	2	5	3	3	8	3	11	–

Table 5
Ablation results on UEA datasets.

Ablation modules	Average accuracy	Average rank
MICOS	0.742	3.5833
MICOS$_{out\_Intra}$	0.730 (−1.62%)	4.0667
MICOS$_{out\_Self}$	0.709 (−4.45%)	4.1667
MICOS$_{out\_TC}$	0.724 (−2.43%)	4.2667
MICOS$_{Cross-Entropy}$	0.691 (−6.87%)	4.2833
MICOS$_{out\_SC}$	0.716 (−3.50%)	4.4833
MICOS$_{self}$	0.648 (−12.67%)	5.0167
MICOS$_{out\_Inter}$	0.666 (−10.24%)	6.1333

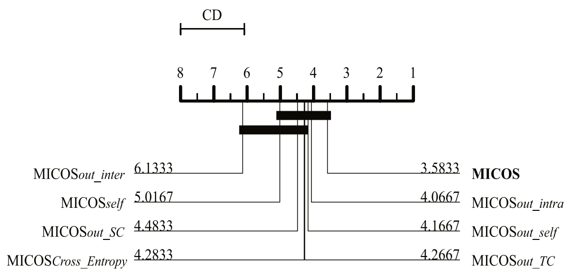

Fig. 7. Critical difference plot of MICOS and its variants on the 30 UEA datasets with $\alpha = 0.05$.

6 participants using a tri-axial accelerometer on the dominant wrist while conducting 4 different activities: EPILEPSY, WALKING, RUNNING, and SAWING [5]. The classification problem diagnoses participants' activities, and 137 test samples are included. For every sample, there are 3 dimensions corresponding to the three axes of the accelerometer. As shown in Fig. 8, we randomly chose one sample for every class from the test set for heatmap visualization. The first row corresponds to the original time series in the Epilepsy dataset. The embeddings of the four samples learned by MICOS are displayed in the corresponding column. The results show that the embeddings of samples from 4 different classes show obvious differences. This demonstrates that the representation learned by MICOS can clearly distinguish the samples of different classes and that the latent class information in the dataset is successfully captured by MICOS.

To dig further, we use the t-SNE algorithm [32] to visualize the learned time series representation embedded in the form of two dimensional images. We compare MICOS with TS2Vec and ShapeNet, which are state-of-the-art methods among current self-supervised and supervised methods, respectively. Fig. 9 shows the test results of the three datasets HandMovementDirection, NATOPS, and JapaneseVowels. The HandMovementDirection dataset contains 147 test samples in 4 different classes. The task is to classify the direction of movement from the Magnetoencephalography (MEG) data recorded during the hand's activity. The NATOPS dataset is used to automatically detect the motion of various naval air training and operating procedures standardization motions used to control plane movements, which contains 180 test samples in 6 different classes. The JapaneseVowels dataset includes recoredings of nine Japanese-male speakers saying the vowels 'a' and 'e'. The classification task predicts the speaker. JapaneseVowels contains 270 test samples in 9 different classes. For simplicity, we uniformly use Arabic numerals to distinguish different categories on different datasets. As shown in Fig. 9, each row represents the test result of a dataset. The first column is a visualization of the original data. The second column is a visualization of the embeddings learned by TS2Vec, and the following columns show the embeddings learned by ShapeNet

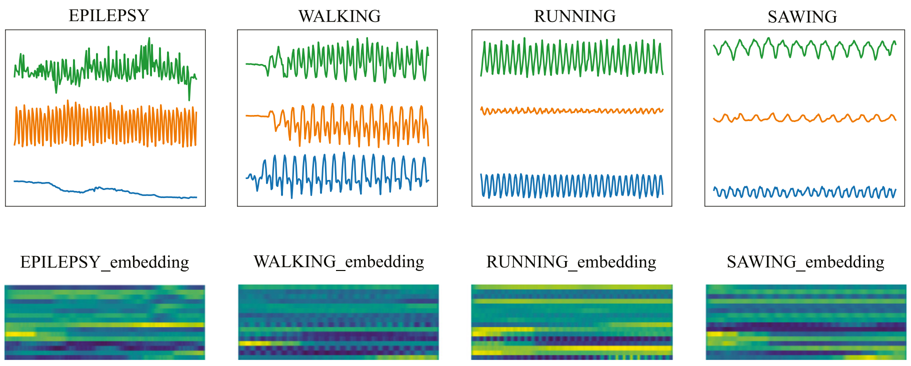

Fig. 8. Heatmap visualization of representations learned by MICOS on the Epilepsy dataset.

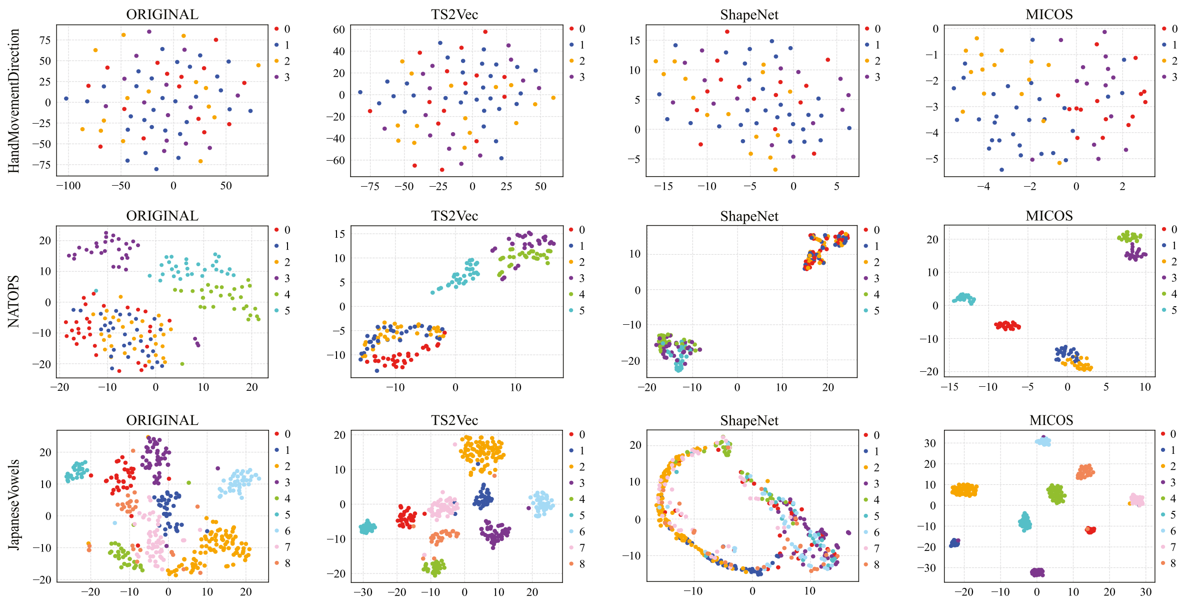

Fig. 9. The t-SNE visualization of the representation space for the datasets HandMovementDirection, NATOPS and JapaneseVowels.

and MICOS. The following conclusion can be drawn from the results: (1) among the three methods, MICOS performs best on all three datasets, which demonstrates the superiority of our method compared to the existing baselines; (2) after encoding by MICOS, the distances between data samples from different classes become larger and the distances between data samples from the same class are smaller, which indicates the effectiveness of the proposed MSCL loss function.

4.6. Runtime analysis

Table 6 shows the runtime (in seconds) in the training and test phases for all the compared deep learning based-methods. We did not consider traditional methods, such as ED and DTW$_I$, because they require different operating environments and have much lower accuracy than MICOS. For simplicity, according to the length of the time series, we picked a short dataset Libras, a medium dataset Epilepsy and a long dataset StandWalkJump from the 30 UEA datasets for experiments. It should be understood that because the numbers of instances in the datasets are also different, the running time of these methods on these datasets does not strictly increase with the length of the time series. For all the compared methods, we follow the hyperparameters as described by the authors. When the running time is less than 0.1 s, we uniformly mark it "<0.1". The results show that our method achieved an acceptable inference time (less than 3 s) compared with the other methods, and it runs faster in the training process than the best baseline ShapeNet and the self-supervised T-Loss. Compared to methods such as TapNet, MLSTM-FCN, and TS2Vec, MICOS is slower due to the MSCL being time-consuming, which is a trade-off between accuracy and efficiency.

5. Conclusion

In this paper, we present a novel mixed supervised contrastive learning framework named MICOS for MTS classification. In particular, based on the two-channel encoder designed to extract complicated spatio-temporal features of MTS, a mixed supervised contrastive loss calculated at the timestamp level with data labels is proposed. The experimental results demonstrate that MICOS can achieve better performance than 13 state-of-the-art methods. Moreover, the ablation study shows the effectiveness of the proposed components. Our future work will be oriented toward extending MICOS to support multivariate time series with missing values and unequal lengths in more realistic scenarios.

Table 6
Runtime (in seconds) for each method in the training and test phases.

	Methods	Libras		Epilepsy		StandWalkJump	
		Training	Test	Training	Test	Training	Test
Self-supervised	TNC	11.6	<0.1	4.9	<0.1	4.0	<0.1
	TST	88.8	<0.1	98.4	<0.1	182.6	<0.1
	TS-TCC	88.5	<0.1	78.3	<0.1	83.0	<0.1
	T-LOSS	4303.1	1.3	4654.5	1.1	7204.0	0.3
	TS2Vec	14.3	2.3	21.3	2.6	126.2	0.3
Supervised	MLSTM-FCN	41.4	0.7	49.0	0.7	35.3	0.7
	TapNet	76.0	<0.1	212.8	<0.1	245.5	<0.1
	TS2Vec (Supervised)	16.1	2.4	20.0	2.5	330.5	0.1
	ShapeNet	1247.5	1500.0	186204.2	4200.7	2217.2	1131.0
	MICOS	1177.7	2.7	2211.6	2.9	602.9	0.9

CRediT authorship contribution statement

Shilei Hao: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Data Curation, Writing – original draft, Writing – review & editing, Visualization. **Zhihai Wang:** Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Afanasiev D. Alexander:** Resources, Writing – review & editing. **Jidong Yuan:** Formal analysis, Resources, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Wei Zhang:** Software, Resources, Writing – review & editing.

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

We have shared the link to our code in the paper.

Acknowledgments

This work was supported by the National Key R&D Program of China (No. 2022YFE0200400), the Beijing Natural Science Foundation (No. 4202029, 4214067), the Beijing Excellent Talents (No. 2017000020124G056), and the Fundamental Research Funds for the Central Universities (No. 2022JBMC011).

References

[1] G. Li, B. Choi, J. Xu, S.S. Bhowmick, K.-P. Chun, G.L. Wong, ShapeNet: A shapelet-neural network approach for multivariate time series classification, in: Proceedings of the 35th AAAI Conference on Artificial Intelligence, 2021, pp. 8375–8383.
[2] A. Stevner, D. Vidaurre, J. Cabral, K. Rapuano, S.F.V. Nielsen, E. Tagliazucchi, H. Laufs, P. Vuust, G. Deco, M.W. Woolrich, et al., Discovery of key whole-brain transitions and dynamics during human wakefulness and non-rem sleep, Nature Commun. 10 (1) (2019) 1–14.
[3] H. Yang, P. Li, H. Li, An oil imports dependence forecasting system based on fuzzy time series and multi-objective optimization algorithm: Case for China, Knowl.-Based Syst. 246 (2022).
[4] Z. Yue, Y. Wang, J. Duan, T. Yang, C. Huang, Y. Tong, B. Xu, TS2vec: Towards universal representation of time series, in: Proceedings of the 36th AAAI Conference on Artificial Intelligence, 2022.
[5] A. Bagnall, H.A. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, E. Keogh, The UEA multivariate time series classification archive, 2018, 2018, arXiv preprint arXiv:1811.00075.
[6] P. Schäfer, U. Leser, Multivariate time series classification with weasel+muse, 2017, arXiv preprint arXiv:1711.11343.
[7] X. Zhang, Y. Gao, J. Lin, C.-T. Lu, TapNet: Multivariate time series classification with attentional prototypical network, in: Proceedings of the 34th AAAI Conference on Artificial Intelligence, 2020, pp. 6845–6852.
[8] J.-Y. Franceschi, A. Dieuleveut, M. Jaggi, Unsupervised scalable representation learning for multivariate time series, in: Proceedings of the 33rd Conference on Neural Information Processing Systems, 2019, pp. 4652–4663.
[9] B. Ghaddar, J. Naoum-Sawaya, High dimensional data classification and feature selection using support vector machines, European J. Oper. Res. 265 (3) (2018) 993–1004.
[10] E. Eldele, M. Ragab, Z. Chen, M. Wu, C.K. Kwoh, X. Li, C. Guan, Time-series representation learning via temporal and contextual contrasting, 2021, arXiv preprint arXiv:2106.14112.
[11] P. Khosla, P. Teterwak, C. Wang, A. Sarna, Y. Tian, P. Isola, A. Maschinot, C. Liu, D. Krishnan, Supervised contrastive learning, in: Proceedings of the 34th Conference on Neural Information Processing Systems, 2020.
[12] A.P. Ruiz, M. Flynn, J. Large, M. Middlehurst, A. Bagnall, The great multivariate time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 35 (2) (2021) 401–449.
[13] S. Seto, W. Zhang, Y. Zhou, Multivariate time series classification using dynamic time warping template selection for human activity recognition, in: Proceedings of the 2015 IEEE Symposium Series on Computational Intelligence, 2015.
[14] M. Shokoohi-Yekta, J. Wang, E. Keogh, On the non-trivial generalization of dynamic time warping to the multi-dimensional case, in: Proceedings of the 2015 SIAM International Conference on Data Mining, 2015, pp. 289–297.
[15] M.S. Cetin, A. Mueen, V.D. Calhoun, Shapelet ensemble for multidimensional time series, in: Proceedings of the 2015 SIAM International Conference on Data Mining, 2015, pp. 307–315.
[16] W. Pei, H. Dibeklioğlu, D.M. Tax, L. van der Maaten, Multivariate timeseries classification using the hidden-unit logistic model, IEEE Trans. Neural Netw. Learn. Syst. 29 (4) (2017) 920–931.
[17] M.G. Baydogan, G. Runger, Learning a symbolic representation for multivariate time series classification, Data Min. Knowl. Discov. 29 (2) (2015) 400–422.
[18] H. Li, Accurate and efficient classification based on common principal components analysis for multivariate time series, Neurocomputing 171 (2016) 744–753.
[19] Y. Zheng, Q. Liu, E. Chen, Y. Ge, J.L. Zhao, Time series classification using multi-channels deep convolutional neural networks, in: Proceedings of the 15th International Conference on Web-Age Information Management, 2014, pp. 298–310.
[20] Y. Hao, H. Cao, A new attention mechanism to classify multivariate timeseries, in: Proceedings of the 29th International Joint Conference on Artificial Intelligence, 2020, pp. 1999–2005.
[21] S. Hochreiter, J. Schmidhuber, Long short-term memory, Neural Comput. 9 (8) (1997) 1735–1780.
[22] F. Karim, S. Majumdar, H. Darabi, S. Harford, Multivariate lstm-fcns for time series classification, Neural Netw. 116 (2019) 237–245.
[23] T. Gao, X. Yao, D. Chen, Simcse: Simple contrastive learning of sentence embeddings, 2021, arXiv preprint arXiv:2104.08821.
[24] T. Duboudin, E. Dellandréa, C. Abgrall, G. Hénaff, L. Chen, Encouraging intra-class diversity through a reverse contrastive loss for single-source domain generalization, in: Proceedings of the 18th IEEE/CVF International Conference on Computer Vision, 2021, pp. 51–60.
[25] X. Zhao, R. Vemulapalli, P.A. Mansfield, B. Gong, B. Green, L. Shapira, Y. Wu, Contrastive learning for label efficient semantic segmentation, in: Proceedings of the 18th IEEE/CVF International Conference on Computer Vision, 2021, pp. 10623–10633.
[26] Q. Xu, A. Baevski, T. Likhomanenko, P. Tomasello, A. Conneau, R. Collobert, G. Synnaeve, M. Auli, Self-training and pre-training are complementary for speech recognition, in: Proceedings of the 2021 IEEE International Conference on Acoustics, Speech and Signal Processing, IEEE, 2021, pp. 3030–3034.

[27] S. Tonekaboni, D. Eytan, A. Goldenberg, Unsupervised representation learning for time series with temporal neighborhood coding, 2021, arXiv preprint arXiv:2106.00750.
[28] G. Zerveas, S. Jayaraman, D. Patel, A. Bhamidipaty, C. Eickhoff, A transformer-based framework for multivariate time series representation learning, in: Proceedings of the 27th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2021, pp. 2114–2124.
[29] D. Hendrycks, K. Gimpel, Gaussian error linear units (gelus), 2016, arXiv preprint arXiv:1606.08415.
[30] J. Demšar, Statistical comparisons of classifiers over multiple data sets, J. Mach. Learn. Res. 7 (2006) 1–30.
[31] J. Yuan, Q. Lin, W. Zhang, Z. Wang, Locally slope-based dynamic time warping for time series classification, in: Proceedings of the 28th ACM International Conference on Information and Knowledge Management, 2019, pp. 1713–1722.
[32] L. Van der Maaten, G. Hinton, Visualizing data using t-sne, J. Mach. Learn. Res. 9 (11) 2579–2605.