Knowledge-Based Systems 245 (2022) 108606

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

TimeCLR: A self-supervised contrastive learning framework for univariate time series representation

Xinyu Yang, Zhenguo Zhang [*], Rongyi Cui

Department of Computer Science and Technology, Yanbian University, 977 Gongyuan Road, Yanji, 133002, P.R. China

A R T I C L E   I N F O

Article history:
Received 28 September 2021
Received in revised form 25 February 2022
Accepted 14 March 2022
Available online 21 March 2022

Keywords:
Univariate Time series
Representation learning
Self-supervised learning
Contrastive learning
Data augmentation

A B S T R A C T

Time series are usually rarely or sparsely labeled, which limits the performance of deep learning models. Self-supervised representation learning can reduce the reliance of deep learning models on labeled data by extracting structure and feature information from unlabeled data and improve model performance when labeled data is insufficient. Although SimCLR has achieved impressive success in the computer vision field, direct applying SimCLR to time series field usually performs poorly due to the part of data augmentation and the part of feature extractor not being adapted to the temporal dependencies within the time series data. In order to obtain high-quality time series representations, we propose TimeCLR, a framework which is suitable for univariate time series representation, by combining the advantages of DTW and InceptionTime. Inspired by the DTW-based k-nearest neighbor classifier, we first propose the DTW data augmentation that can generate DTW-targeted phase shift and amplitude change phenomena and retain time series structure and feature information. Inspired by the current state-of-the-art deep learning-based time series classification method, InceptionTime, which has good feature extraction capabilities, we designed a feature extractor capable of generating representations in an end-to-end manner. Finally, combining the advantages of DTW data augmentation and InceptionTime, our proposed TimeCLR method successfully extends SimCLR and applies it to the time series field. We designed a variety of experiments and performed careful ablation studies. Experimental results show that our proposed TimeCLR method can not only achieve comparable performance to supervised InceptionTime on multiple tasks, but also produce better performance than supervised learning models in the case of insufficient labeled data, and can be flexibly applied to univariate time series data from different domains.

1. Introduction

Time series data are collected progressively every day from sensors in the medical, financial, industrial and other domains, its main characteristic is the strong temporal dependencies within the data. Deep learning has become a widely used method in the field of time series analysis and has been successfully applied to time series in different domains [1,2]. However, time series data are rarely or sparsely labeled [3]. Different from images and text data, real-world time series are high-dimensional and complex, often do not have features that can be recognized directly by humans, or only have features that can be recognized by specialists in a particular domain, which makes it impossible to directly label time series, or makes it very expensive to label time series data. While deep learning is a data-driven approach, the lack of labeled data limits the performance of deep learning models. One way to address this problem is to learn structure and feature information from unlabeled time series to compose representations which reduce the reliance on expensive manual labeling.

Self-supervised representation learning extracts effective representations from unlabeled data for downstream tasks through self-generated signals. In the field of computer vision, self-supervised representation learning has recently received a lot of attention. Self-supervised representation learning can yield comparable performance to supervised trained models in this field [4,5]. The self-generated signals are derived from different pretext tasks on image [6,7]. One of the successful pretext tasks is to use contrastive learning to learn invariant representations from different augmented image views [8,9,4], where the most widely used method is SimCLR.

A heuristic way of time series self-supervised representation learning is to apply SimCLR directly to the time series field. Unfortunately, this way does not work well on time series data. First, time series data has strong temporal dependencies, traditional time series data augmentations may disrupt the temporal dependencies within the data, thus impairing the structure and feature

[*] Corresponding author.
E-mail addresses: 2020050049@ybu.edu.cn (X. Yang), zgzhang@ybu.edu.cn (Z. Zhang), cuirongyi@ybu.edu.cn (R. Cui).

https://doi.org/10.1016/j.knosys.2022.108606
0950-7051/© 2022 Elsevier B.V. All rights reserved.

information of time series. Second, some data augmentation operations that can produce great results on images are not well fitted to time series. Third, feature extractors with excellent performance on images do not extract the structure and feature information of time series well. Mohsenvand et al. [10] applied the extended SimCLR to EEG time series, but this method is designed for specific application scenarios and cannot be generalized to other time series domains.

To address the above issues and apply SimCLR to the time series field, we propose a univariate **Time** series self-supervised **C**ontrastive **L**earning framework for **R**epresentation (TimeCLR). First, as a verified method to compare the similarity between univariate time series well, Dynamic Time Warping (DTW) targets the phase shifts and amplitude changes of time series so that the time series are warped in a non-linear manner to match each other [11,12]. The better performance produced by the DTW-based k-nearest neighbor algorithm on time series classification problems [13–15] suggests that phase shifts and amplitude shifts are likely to exist for time series of the same class. Inspired by these, we propose a novel univariate time series data augmentation method named DTW data augmentation, which has a special self-supervised trained autoencoder. In which, the encoder can map the univariate time series to an embedding vector in the Euclidean feature space, the movement of the embedding vector can cause the phase shift and amplitude change of the time series, and the decoder can reconstruct the embedding vector back to the time series. DTW data augmentation not only can produce phase shifts and amplitude changes, but also can retain the structure and feature information of the time series. Second, our proposed self-supervised contrastive learning framework enables the feature extractor to learn the invariant structure and feature information in both views by minimizing the similarity of two views from the same sample after two separate data augmentation operations, which thus can generate high-quality representations. InceptionTime, as the state-of-the-art supervised deep learning method on time series classification problem [16], shows that it has good structure and feature information extraction capability for time series data, inspired by this, we design the feature extractor in TimeCLR combining the advantages of InceptionTime. Third, our multiple experiments on supervised learning, semi-supervised learning, transfer learning and clustering tasks show that TimeCLR can achieve comparable performance with supervised InceptionTime and can generalize across different time series domains. We also designed various ablation experiments to verify the important role of each component of TimeCLR.

Our contributions can be summarized as follows:

- We propose a novel method for univariate time series data augmentation named DTW data augmentation that not only generates phase shifts and amplitude changes, but also retains the structure and feature information of the time series.
- We design a feature extractor that can generate univariate time series representations in an end-to-end manner, drawing on the advantages of InceptionTime, the current state-of-the-art deep learning-based time series classification method.
- Combining the advantages of DTW data augmentation and InceptionTime model, we successfully applied extended SimCLR to the time series field.
- We conducted multiple types of experiments and the results showed the effectiveness and generality of our method.

The rest of this paper is organized as follows. In the next section, we discuss the related works. Section 3 gives the preliminaries and notations used in this paper. In Section 4, we describe the proposed method in detail, including the DTW data augmentation and the self-supervised contrastive learning framework. The experimental setup and results are presented in Section 5. Finally, we conclude the paper in Section 6.

2. Related work

2.1. Time series modeling

Shapelets-based methods are an important category of time series classification methods. These methods extract discriminative subsequences from the training set and feed them into an off-the-shelf classifier [17–19]. Although the training complexity of extracting shapelets is too large to be applied to larger data sets, the success of this category of methods suggests that the key to time series modeling is to extract structure and feature information from the time series. Distance-based methods rely on k-Nearest Neighbor classifiers based on Euclidean distance or DTW for classification [11,20]. The success of the DTW-based k-nearest neighbor algorithm indicates the existence of phase shifts and amplitude changes in the same class of time series data. Cai et al. [21] leverage DTW to design a novel component of neural network to obtain better feature extraction, showing that DTW is beneficial for improving the feature extraction of neural networks. More recently, much attention has been paid to deep learning-based methods which use deep neural networks to model time series in an end-to-end manner [22,14,1]. Common neural network architectures for modeling time series can be divided into multilayer perceptron based neural networks, CNN based neural networks and RNN based neural networks. CNNs produce better results than RNNs and MLPs overall, InceptionTime is the current state-of-the-art model, ResNet is the suboptimal model.

2.2. Data augmentation for time series

The purpose of data augmentation is to provide new time series samples. Traditional time series data augmentation methods include Jitter, Scaling, MagWarp, TimeWarp, Permutation, Inversion and Rotation [23]. Where rotation can only be applied to time series data generated by domain-specific sensors. In the literature, many data augmentation methods focus on time series of one domain. In order to augment the ECG and EEG datasets, Haradal et al. [24] used generative adversarial networks for the generation and discrimination of synthetic biosignals. Ramponi et al. [25] performed data augmentation of noisy time series using a conditional generative adversarial network with irregular sampling. Cao et al. [26] performed data augmentation on electrical signal recordings using samples belonging to different classes. In addition, some studies generate new time series samples by interpolation of two data. Interpolation is very difficult due to the frequent nonlinear transformation in time scale [27]. DeVries et al. [28] showed that interpolation in the deep feature space is superior to direct interpolation of time series. This type of method is not adapted to our contrastive learning framework, because two samples are required to generate one data. Our method requires time series data augmentations that are capable of retaining information about the structure and feature of the time series, and can be generalized to different domains.

2.3. Self-supervised representation learning for time series

Self-supervised Representation Learning is receiving more and more attention due to its great success in the field of computer vision [6,29,30]. The one which has received the most attention is the SimCLR proposed by Chen et al. [4]. Self-supervised representation Learning for time series is becoming more and more popular recently. The method proposed by Hyvarinen et al. extracts the representation by using contrastive loss to predict the segment ID of time series [31]. The method proposed by

Franceschi et al. uses negative sampling based on temporal relationships and triplet loss to extract scalable representations of time series [32]. The method proposed by Jawed et al. utilizes the prediction task as a pretext task to generate representations [33]. Tonekaboni et al. [34] learning generalized representations for nonstationary time series using the relationship of temporal neighborhoods. Fan et al. [35] learning representations by exploring time series inter-sample relationships and intra-temporal relationships. Eldele et al. [36] learning time series representation via temporal and contextual contrasting. **The best results achieved in these self-supervised representation learning methods for time series are comparable to the corresponding supervised models.** In addition, Anand et al. [37] generates representations for time series using visual perception and unsupervised triple loss training, such representations are mainly used for clustering. Autowarp [38] can also generate latent representations in the process of unsupervised learning of time series metrics, but such representations are mainly used to obtain high-quality metrics. There are also some studies for domain-specific time series representation. SSL-ECG learns ECG time series representations by applying six transformations to the dataset [39]. SEQCLR designed EEG-related data augmentations and extended the SimCLR to EEG time series [10], but its data augmentation operations may corrupt the time correlation within the time series and it is limited to applying to EEG data only. Our proposed TimeCLR not only yields comparable performance to the state-of-the-art supervised deep learning model InceptionTime, but can also be applied to time series data from different domains.

3. Preliminaries and notations

For the convenience of illustrating our method, we first give the following notations.

3.1. Time series

A univariate time series $\boldsymbol{X} = [x^1, x^2, \ldots, x^L]$ is an ordered set of L real-valued observed variables, thus $\boldsymbol{X} \in \mathbb{R}^L$. The data points $x^1, x^2, \ldots, x^L$ in a time series are usually arranged in temporal order and spaced at equal time intervals. In the following, we simplify the univariate time series as time series.

We denote a time series dataset as $\mathcal{X} = \{\boldsymbol{X_n}\}_{n=1}^{N}$, where N is the dataset size, $\boldsymbol{X_n}$ denotes a time series sample, n is the index number of the time series sample in $\mathcal{X}$. When collecting data, it is usually necessary to truncate, pad or resample the time series samples to ensure that the length of all samples in the dataset is L. In the dataset, some time series samples have been pre-labeled artificially, some are unlabeled, and unlabeled time series data predominate in the real world.

3.2. Dynamic time warping (DTW)

DTW is a reliable tool for comparing time series similarity for possible phase shifts and amplitude changes in time series. When the length of two time series samples $\boldsymbol{X_n}$ and $\boldsymbol{X_m}$ are L, in order to calculate the DTW of these two time series, it is necessary to construct an $L \times L$ alignment matrix $\boldsymbol{M}$. The element M_{ij} of matrix $\boldsymbol{M}$ represents the cost of aligning the i-th observation of $\boldsymbol{X_n}$ to the j-th observation of $\boldsymbol{X_m}$.

$$M_{ij} = \left\| x_n^i - x_m^j \right\|_2^2 \tag{1}$$

A warping path $WP = \left[wp^1, wp^2, \ldots, wp^p, \ldots, wp^P\right]$ $(L \le P < 2L)$ is the set of index pairs formed by aligning each observation of $\boldsymbol{X_n}$ to the observations of $\boldsymbol{X_m}$, where $wp^1 = (1, 1)$ and $wp^P = (L, L)$. Obviously, there are multiple feasible warping paths. To ensure that there is no crossover in the alignment of all observations, wp^{p+1} can only be an element of the set $\{(i+1, j), (i, j+1), (i+1, j+1)\}$ when $wp^p = (i, j)$.

Warping distance wd is the sum of the cost of all aligned pairs in a warping path.

$$wd\,(WP) = \sum_{p=1}^{P} M_{wp^p} \tag{2}$$

DTW is the minimum warping distance generated by all feasible warping paths.

$$DTW\,(\boldsymbol{X_n}, \boldsymbol{X_m}) = argmin_{wd} \left(\sum_{p=1}^{P} M_{wp^p} \right) \tag{3}$$

Since the dynamic programming algorithm for computing DTW needs to build the alignment matrix, the time complexity of the DTW is $O\left(L^2\right)$. But the method proposed by Mueen et al. [40] can achieve the amortized time complexity of computing DTW to $O\,(L)$, therefore the cost of DTW is acceptable.

4. Methodology

In this section, we introduce our proposed TimeCLR framework. We first introduce DTW data augmentation, which utilizes a special self-supervised trained autoencoder that can generate phase shifts and amplitude changes for time series data. Then, we propose the training framework for self-supervised contrastive representation learning for time series data. The DTW data augmentation and the framework enable the feature extractor to produce high-quality representations.

4.1. DTW data augmentation

The DTW-based k-nearest neighbor classifier yields better results on the time series classification problem, which inspires us that the phenomenon of phase shift and amplitude change targeted by DTW may exist in the same class of time series, and the data augmentation that produces this phenomenon may retain the structure and feature information of time series.

It is difficult to directly operate on the observations of the time series to reflect the phase shifts and amplitude changes. To solve this problem, we propose the DTW data augmentation, which uses a specially self-supervised trained encoder to map the time series into the Euclidean feature space. The movement of the embedding vector in the Euclidean feature space can produce phase shifts and amplitude changes.

As shown in Fig. 1, the encoder of the autoencoder is a siamese neural network with two encoder blocks, both of which have the same structure and share weights. The encoder block projects the time series into a high-dimensional Euclidean feature space, forming an embedding vector that retains all information. The decoder of the autoencoder is also a siamese neural network with two decoder blocks, both of which have the same structure and share weights. The decoder block is used to reconstruct the embedding vector of the Euclidean feature space to a time series, and its architecture is symmetric with the encoder block. Fig. 2 illustrates the architecture of the encoder block and decoder block that we use. Note that their architecture can be flexibly changed depending on the length of the time series data and the training results for a specific task. In the following, we describe in detail our special self-supervised training method.

We construct a special dataset $\mathcal{D} = \{[(\boldsymbol{X_{i1}}, \boldsymbol{X_{i2}}), \widehat{y}_i]\}_{i=1}^{N(N-1)/2}$ for self-supervised training of the autoencoder. In each element $[(\boldsymbol{X_{i1}}, \boldsymbol{X_{i2}}), \widehat{y}_i]$ of $\mathcal{D}$, i is the index number of the element in $\mathcal{D}$, $\boldsymbol{X_{i1}}$ and $\boldsymbol{X_{i2}}$ are two different time series taken from the time series dataset $\mathcal{X}$, $\widehat{y}_i$ is the DTW of these two time series. Since the size of

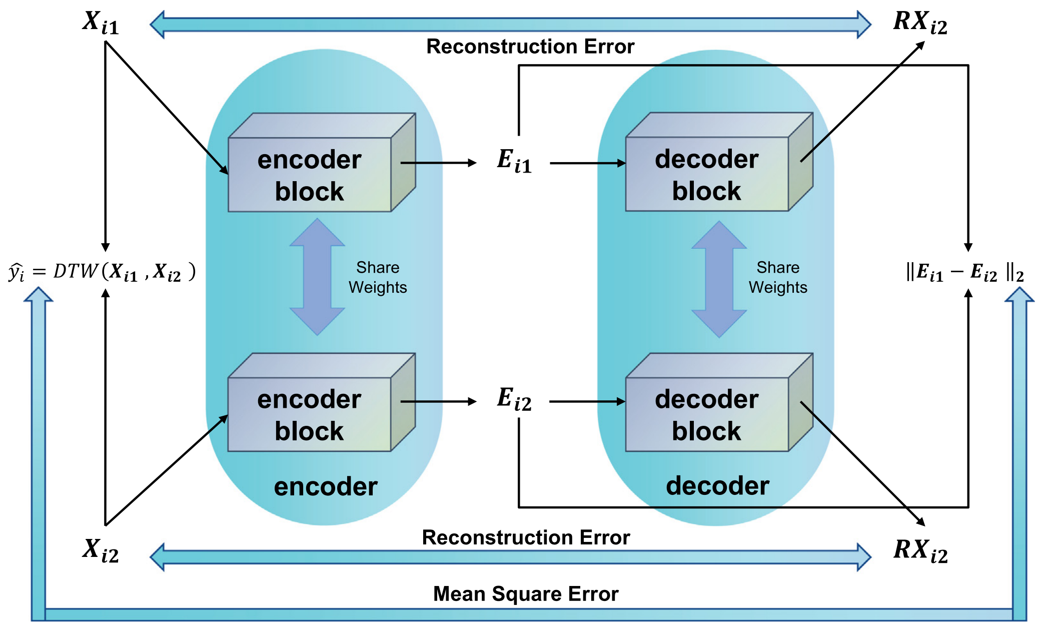

Fig. 1. An overview of the DTW data augmentation training process. The encoder contains two encoder blocks with the same architecture and shared weights. The encoder block is used to project the time series into an embedding vector of the Euclidean feature space. The decoder contains two decoder blocks with the same architecture and shared weights. The decoder block is used to invert the mapping relationship and reconstruct the embedding vector into the corresponding time series.

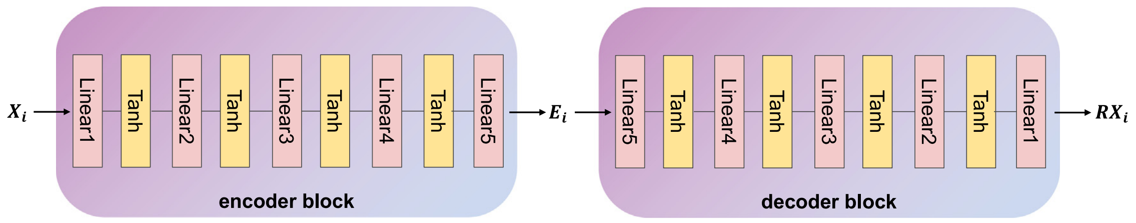

Fig. 2. The architecture of the encoder block and decoder block. Their architectures are flexible to change depending on the length of the time series data and the training results for a specific task.

the time series dataset $\mathcal{X}$ is N, the size of $\mathcal{D}$ is $N(N-1)/2$. If the length of all time series is L, the time complexity of constructing the dataset $\mathcal{D}$ is $O(N^2L^2)$, but the method proposed by Mueen et al. [40] can make it run in $O(N^2L)$. It is worth noting that since time series in $\mathcal{D}$ do not need to be artificially labeled, we could easily add more samples to $\mathcal{D}$ in reality when the existing samples are insufficient. Since the self-supervised training of autoencoder is to learn the mapping relationship between different distances, it is feasible to add time series belonging to other domains but with the same length. When the available samples are sufficient, we can also consider removing some elements of $\mathcal{D}$ to fasten the self-supervised training of autoencoder.

For each element $[(\boldsymbol{X_{i1}}, \boldsymbol{X_{i2}}), \hat{y}_i]$ in $\mathcal{D}$, $\boldsymbol{X_{i1}}$ and $\boldsymbol{X_{i2}}$ are simultaneously input to the encoder to produce the corresponding embedding vector $\boldsymbol{E_{i1}}$ and $\boldsymbol{E_{i2}}$ in high-dimensional Euclidean feature space.

$$\boldsymbol{E_{i1}}, \boldsymbol{E_{i2}} = encoder\,(\boldsymbol{X_{i1}}, \boldsymbol{X_{i2}}) \tag{4}$$

The input of the decoder are two embedding vectors and the output are the corresponding two reconstructed time series. $\boldsymbol{E_{i1}}$ and $\boldsymbol{E_{i2}}$ are reconstructed into time series $\boldsymbol{RX_{i1}}$ and $\boldsymbol{RX_{i2}}$ by the decoder.

$$\boldsymbol{RX_{i1}}, \boldsymbol{RX_{i2}} = decoder\,(\boldsymbol{E_{i1}}, \boldsymbol{E_{i2}}) \tag{5}$$

The loss function when training the autoencoder consists of two parts:

$$\mathcal{L} = \mathcal{L}_{\mathcal{E}} + \mathcal{L}_{\mathcal{D}} \tag{6}$$

where $\mathcal{L}_{\mathcal{E}}$ denotes the loss function for the encoder and $\mathcal{L}_{\mathcal{D}}$ denotes the loss function for the decoder.

Our goal in training the encoder is that the movement of the embedding vector in the Euclidean feature space can produce semantic changes in phase shift and amplitude change for the time series. Therefore, we want to minimize the mean square error of the Euclidean distance between the embedding vector and the *DTW* between the time series. We define $\mathcal{L}_{\mathcal{E}}$ as:

$$\mathcal{L}_{\mathcal{E}} = \frac{1}{|\mathcal{D}|}\sum_i (\|\boldsymbol{E_{i1}} - \boldsymbol{E_{i2}}\|_2 - \hat{y}_i)^2 \tag{7}$$

Our goal in training the decoder is to map the embedding vector from Euclidean feature space back to the time series. Thus we define $\mathcal{L}_{\mathcal{D}}$ to minimize the reconstruction error between the time series and the reconstructed time series:

$$\mathcal{L}_{\mathcal{D}} = \frac{1}{2|\mathcal{D}|}\sum_i (\|\boldsymbol{X_{i1}} - \boldsymbol{RX_{i1}}\|_2)^2 + \frac{1}{2|\mathcal{D}|}\sum_i (\|\boldsymbol{X_{i2}} - \boldsymbol{RX_{i2}}\|_2)^2 \tag{8}$$

The process of training the autoencoder is the process of minimizing the loss function $\mathcal{L}$. We define the parameters of autoencoder as θ. Training the autoencoder is to find the optimal parameters θ^* that minimizes $\mathcal{L}$.

$$\theta^* = argmin_{\theta}\mathcal{L} \tag{9}$$

In practice, we first train the encoder until $\mathcal{L}_{\mathcal{E}}$ converges, then freeze the parameters of the encoder and train the decoder to

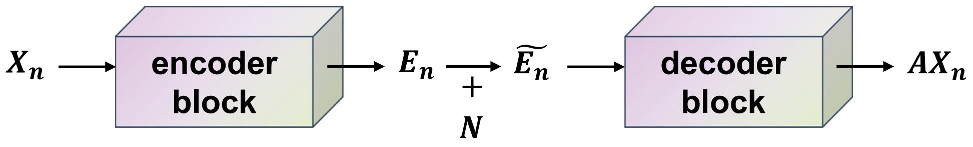

Fig. 3. The process of DTW data augmentation. We first get the embedding vector of the time series by encoder block, then add noise to the embedding vector and reconstruct it into a time series by decoder block.

learn how to map the embedding vector back to the time series. If $\mathcal{L}_{\mathcal{E}}$ is difficult to converge during the training process, we can increase the dimensionality of the embedding vector, switch to a larger model for the autoencoder, and add more samples to the dataset $\mathcal{D}$.

After the training process, we use the encoder block and the decoder block for data augmentation. The movement of the embedding vector in the Euclidean feature space will reflect the phase shift and amplitude change of the time series.

The process of DTW data augmentation is shown in Fig. 3. First, we map a time series $\boldsymbol{X_n}$ into an embedding vector $\boldsymbol{E_n}$ by the encoder block.

$$\boldsymbol{E_n} = encoderblock\left(\boldsymbol{X_n}\right) \tag{10}$$

Next, we add a noise vector $\boldsymbol{N}$ to $\boldsymbol{E_n}$ to form $\tilde{\boldsymbol{E_n}}$. Each dimension of $\boldsymbol{N}$ is randomly drawn from a normal distribution with mean 0 and standard deviation S_{noise}.

$$\tilde{\boldsymbol{E_n}} = \boldsymbol{E_n} + \boldsymbol{N} \tag{11}$$

The above operations performed in the Euclidean feature space represent phase shifts and amplitude changes on the time series. Finally, we map $\tilde{\boldsymbol{E_n}}$ back to the time series by the decoder block to get the time series $\boldsymbol{AX_n}$ after data augmentation.

$$\boldsymbol{AX_n} = decoderblock\left(\tilde{\boldsymbol{E_n}}\right) \tag{12}$$

4.2. Self-supervised contrastive learning framework

Our self-supervised contrastive learning aims to obtain a feature extractor to convert time series to representation vectors that preserve inherent structure and feature information. Similar to SimCLR [4], our method perform contrastive learning by maximizing the similarity of two augmented views of the same time series through a contrastive loss in the latent space.

As shown in Fig. 4, we first apply two separate data augmentations to the sample $\boldsymbol{X_n}$ in the dataset $\mathcal{X}$ to obtain two views $\tilde{\boldsymbol{X}}_{\boldsymbol{n}}^{(1)}$ and $\tilde{\boldsymbol{X}}_{\boldsymbol{n}}^{(2)}$. The DTW data augmentation proposed in Section 4.1 is essential to produce high-quality time series representations but we also have the option of using the data augmentation Jitter, Scaling, MagWarp, TimeWarp, Permutation, and Inversion proposed by [23]. For the convenience of illustration, we refer to the data augmentation methods proposed by [23] as traditional time series data augmentations. In practice, we have to decide whether we can use traditional time series data augmentations based on validation performance.

We assume that the time series after data augmentation retains the structure and feature information although it changes in form. Therefore we expect that the feature extractor can learn the structure and feature information of the time series to form a high-quality representation. In our framework, the two feature extractors are two components of a siamese neural network, both feature extractors have the same structure and share weights. Inspired by [5], a larger feature extractor with better feature extraction capability would be more beneficial to obtain a better representation. Since InceptionTime [16] produces outstanding performance on time series classification problems, we borrow the advantages of InceptionTime to design the feature extractor as shown in Fig. 5. InceptionTime is an ensemble of 5 networks in [16], where the feature extractor in our framework uses only one of them. To facilitate flexible adjustment of the dimensionality of the representations, we add a fully-connected layer at the tail of the neural network. In the following, we will call our feature extractor as InceptionTime.

The two views $\tilde{\boldsymbol{X}}_{\boldsymbol{n}}^{(1)}$ and $\tilde{\boldsymbol{X}}_{\boldsymbol{n}}^{(2)}$ are input to the InceptionTime respectively to get the corresponding representations $\boldsymbol{R}_{\boldsymbol{n}}^{(1)}$ and $\boldsymbol{R}_{\boldsymbol{n}}^{(2)}$.

$$\begin{aligned}\boldsymbol{R}_{\boldsymbol{n}}^{(1)} &= InceptionTime\left(\tilde{\boldsymbol{X}}_{\boldsymbol{n}}^{(1)}\right)\\ \boldsymbol{R}_{\boldsymbol{n}}^{(2)} &= InceptionTime\left(\tilde{\boldsymbol{X}}_{\boldsymbol{n}}^{(2)}\right)\end{aligned} \tag{13}$$

Next, the projection head as shown in Fig. 6 projects $\boldsymbol{R}_{\boldsymbol{n}}^{(1)}$ and $\boldsymbol{R}_{\boldsymbol{n}}^{(2)}$ into the latent space where the contrastive loss is applied to obtain $\boldsymbol{Z}_{\boldsymbol{n}}^{(1)}$ and $\boldsymbol{Z}_{\boldsymbol{n}}^{(2)}$, respectively. Projection head is a simple neural network with two linear layers and a ReLU activation function. The projection heads we use are also two components of a siamese neural network, they have the same structure and share weights.

$$\begin{aligned}\boldsymbol{Z}_{\boldsymbol{n}}^{(1)} &= projectionhead\left(\boldsymbol{R}_{\boldsymbol{n}}^{(1)}\right)\\ \boldsymbol{Z}_{\boldsymbol{n}}^{(2)} &= projectionhead\left(\boldsymbol{R}_{\boldsymbol{n}}^{(2)}\right)\end{aligned} \tag{14}$$

For the self-supervised contrastive learning task, we treat the two views obtained by augmenting the same time series as a positive pair example. Instead of generating negative pair examples explicitly, we treat all view pairs that do not belong to the same time series as negative pair examples. For a time series sample $\boldsymbol{X_n}$, we define its loss function as:

$$\mathcal{K}_n = -\log \frac{exp\left[s\left(\boldsymbol{Z}_{\boldsymbol{n}}^{(1)}, \boldsymbol{Z}_{\boldsymbol{n}}^{(2)}\right)/\tau\right]}{\Sigma_{t\neq n} exp\left[s\left(\boldsymbol{Z}_{\boldsymbol{n}}^{(1)}, \boldsymbol{Z_t}\right)/\tau\right] + \Sigma_{t\neq n} exp\left[s\left(\boldsymbol{Z_t}, \boldsymbol{Z}_{\boldsymbol{n}}^{(2)}\right)/\tau\right]} \tag{15}$$

where τ denotes a temperature hyperparameter and $s\left(\boldsymbol{u}, \boldsymbol{v}\right)$ denotes the cosine similarity between $\boldsymbol{u}$ and $\boldsymbol{v}$. The loss function of the self-supervised contrastive learning task is the sum of the losses of each sample. This loss function has been successfully applied in multiple previous works [41,7,42,4,43].

$$\mathcal{K} = \Sigma_{n=1}^{N} \mathcal{K}_n \tag{16}$$

We let ϕ be the set of parameters of the feature extractor InceptionTime and the projection head, and the goal of learning is to find the optimal parameters ϕ^* that minimizes $\mathcal{K}$.

$$\phi^* = argmin_{\phi} \mathcal{K} \tag{17}$$

After training, we discard the data augmentation part and the projection head and only use the feature extractor InceptionTime. The feature extractor is able to convert the time series into the corresponding representations at this point and we can apply the representation to the downstream tasks.

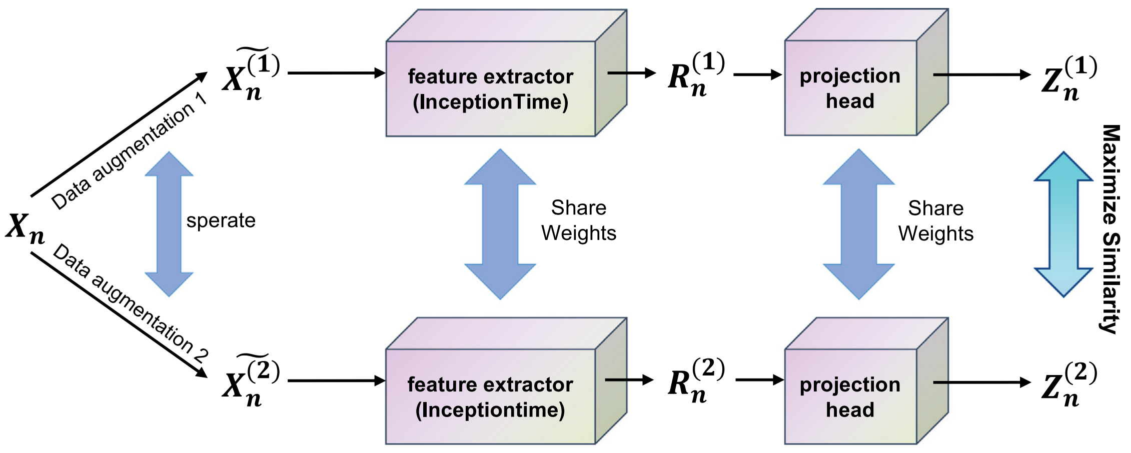

Fig. 4. An overview of the process of self-supervised contrastive learning. Our method perform contrastive learning by maximizing the similarity of two augmented views of the same time series through a contrastive loss in the latent space. The "Data augmentation 1" and "Data augmentation 2" in the figure represent two separate data augmentations.

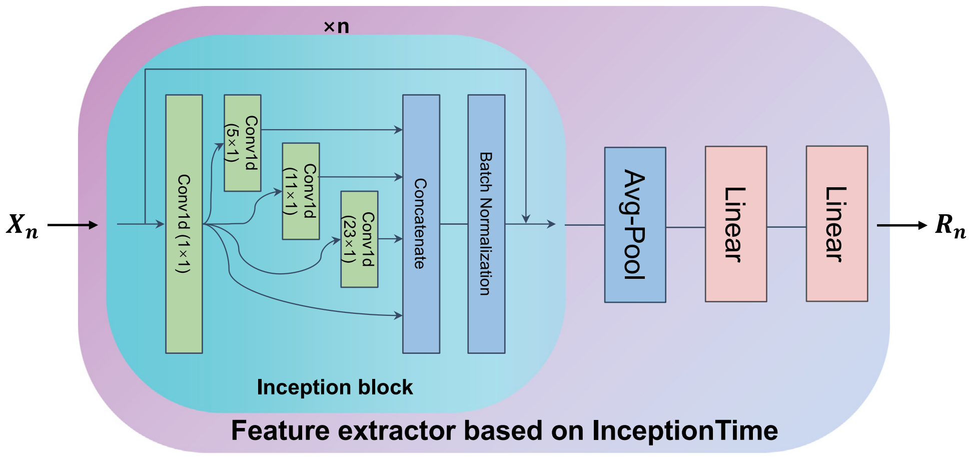

Fig. 5. The architecture of the feature extractor based on InceptionTime.

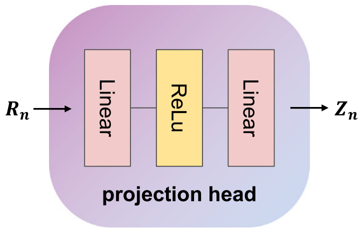

Fig. 6. The architecture of the projection head. The projection head maps the representation $\boldsymbol{R_n}$ to a vector $\boldsymbol{Z_n}$ in the latent space where contrastive loss is applied.

5. Experiments

In this section, we demonstrate the efficacy of TimeCLR through various experiments and verify the necessity of each component of our proposed model through various ablation experiments.

5.1. Experiment settings

5.1.1. Datasets

We conduct experiments on 10 time series datasets. These datasets are all derived from finger bone images [44] and extracted by Cao et al. [45]. All datasets can be divided into 3 domains (Distal, Middle and Proximal), as each dataset is produced from a specific finger bone. Also, they can be divided into 3 types of classification tasks (AgeGroup, Correct and TW) according to the type of task. We can design transfer learning tasks using datasets from different domains but belonging to the same type of classification task. The number of observations for the time series in each of these datasets is 80, and the details of each dataset are shown in Table 1. Each dataset has been pre-segmented into a training set and a testing set.

5.1.2. Evaluation

We chose classification as the downstream task to evaluate the quality of the representation. We follow the standard linear evaluation scheme [42,4]. Specifically, we train a linear classifier with softmax operation on a frozen pre-trained feature extractor (InceptionTime). In our experiments, we evaluate the performance using two metrics, Accuracy and F1-score. When the number of classes in the dataset is greater than 2, we replace the F1-score with the macro-averaged F1-score. In the experiments, we record the optimal results obtained by training from scratch by three times.

5.1.3. Baselines

- **SimCLR**: The self-supervised contrastive learning training framework in this baseline is the same as TimeCLR, with the only difference that only traditional time series data

Table 1
Details of the time series datasets used in our experiments.

Datasets	Abbreviation	Number of classes	Size of training set	Size of testing set
DistalPhalanxOutlineAgeGroup	D.A.G	3	400	139
DistalPhalanxOutlineCorrect	D.C	2	600	276
DistalPhalanxTW	D.TW	6	400	139
MiddlePhalanxOutlineAgeGroup	M.A.G	3	400	154
MiddlePhalanxOutlineCorrect	M.C	2	600	291
MiddlePhalanxTW	M.TW	6	399	154
PhalangesOutlinesCorrect	Pha.C	2	1800	858
ProximalPhalanxOutlineAgeGroup	Pro.A.G	3	400	205
ProximalPhalanxOutlineCorrect	Pro.C	2	600	291
ProximalPhalanxTW	Pro.TW	6	400	205

augmentations can be used. To better adapt SimCLR to the time series field, we use the InceptionTime-based feature extractor by default. It represents the results generated by applying SimCLR directly to the time series domain.

- **Supervised Model**: In this baseline, the InceptionTime model followed by the linear classifier will be trained from scratch. The best results achieved by existing time series self-supervised representation methods are close or comparable to their corresponding supervised models [32,34–36]. Since InceptionTime is the state-of-the-art method for supervised deep learning models on time series classification problems, **this baseline can also represent the results obtained from other state-of-the-art self-supervised time series representation methods.**

5.1.4. Hyperparameter settings and training details

For training the autoencoder used in the DTW data augmentation, we use the *Adam* optimizer with a learning rate of $1e-4$ to perform the mini-batch stochastic gradient descent (*SGD*) with a batch size of 128. We perform 100 epochs when training the encoder and 50 epochs when training the decoder, because the loss function has already converged with this setting. When applying the self-supervised trained autoencoder for DTW data augmentation, we set S_{noise} to 4.

For the training process of self-supervised contrastive learning for representation, we use the *Adam* optimizer with a learning rate of $3e-4$ to perform the *SGD* with a batch size of 128. Our training process is executed for 200 epochs. In the experiment we set the temperature hyperparameter τ to 0.2.

For linear evaluation, we use the *Adam* optimizer with a learning rate of $1e-3$ to perform the *SGD* with a batch size of 128. This process runs for 80 epochs, as we find that further training does not improve the performance.

Our experiments are conducted on a computer equipped with an Inter(R) Xeon(R) Gold 6128 CPU and a NVIDIA GeForce RTX 2080 Ti GPU.

5.2. Comparison with baseline methods

In this section, we compare the TimeCLR with baseline methods. We start with the special self-supervised training process of the autoencoder used in the DTW data augmentation operation with all training data. Then we perform self-supervised contrastive learning using all the training data to obtain a pretrained feature extractor. After that, we freeze the parameters of the feature extractor and train a linear classifier on the pretrained feature extractor using all the training data for linear evaluation.

We test the TimeCLR in two cases. The first case is when we perform self-supervised contrastive learning, each separate data augmentation operation is randomly selected from the six traditional time series data augmentations and the DTW data augmentation. In the second case, each separate data augmentation operation is the DTW data augmentation. But we need to note that the result of each DTW data augmentation operation is different due to the different noise vectors randomly generated. Table 2 shows the experimental results, where the column "TimeCLR(Traditional+DTW)" represents the first case and the column "TimeCLR(DTW)" represents the second case.

Experimental results show that our proposed TimeCLR significantly outperforms the case that directly applies SimCLR to the time series field. This result also validates the importance of our proposed DTW data augmentation for the self-supervised contrastive learning framework. The experimental results illustrate that DTW data augmentation can retain the structure and feature information of the time series and improve the quality of the representation, while the traditional time series data augmentations may impair the structure and feature information of time series. Moreover, we notice that sometimes the "TimeCLR(Traditional+DTW)" case yields better performance, and sometimes the "TimeCLR(DTW)" case yields better performance. Therefore, we should choose whether to add traditional time series data augmentations based on validation performance in practice. In the following experiments, we show the results for both cases. In addition, TimeCLR achieves a performance comparable to the supervised Inceptiontime.

5.3. Semi-supervised learning experiments

We conduct semi-supervised learning experiments for four cases: 75% label, 50% label, 25% label, and 10% label, respectively. For the case of "75% label", we randomly select 75% of the samples from the training set to form the labeled set, and remove the labels from the remaining samples in the training set to form the unlabeled set. We first train the autoencoder used in the DTW data augmentation using all samples in the labeled set and unlabeled set in a self-supervised manner, and then execute self-supervised contrastive learning with all samples in the labeled set and unlabeled set as well, since self-supervised learning is not required for samples to be labeled or unlabeled. Finally, we use the labeled set for linear evaluation or supervised learning. The rest of the cases are similar to the "75% label" case. Note that all self-supervised learning processes do not care whether the data are labeled or not.

The experimental results are shown in Tables 3 and 4. We observe that our proposed TimCLR yields comparable performance to the supervised InceptionTime model and significantly outperforms SimCLR when a larger percentage of labeled data is available. TimeCLR significantly outperforms other baseline methods when the percentage of labeled data is less than 50%. These results reconfirm that time series augmented by DTW data augmentation can retain the structure and feature information. Meanwhile, these results also reflect the importance of DTW data augmentation for the self-supervised contrastive learning process of TimeCLR.

Table 2
Comparison results of our proposed TimeCLR against baselines. "TimeCLR(Traditional+DTW)" represents each separate data augmentation operation used in the self-supervised contrastive learning is randomly selected from the six traditional time series data augmentations and the DTW data augmentation. "TimeCLR(DTW)" represents each separate data augmentation operation is DTW data augmentation.

Datasets	Metrics	TimeCLR (Traditional+DTW)	TimeCLR (DTW)	SimCLR	Supervised
D.A.G	ACC (%)	**74.10**	72.66	70.50	72.66
	F1 (%)	**73.88**	70.75	64.83	71.65
D.C	ACC (%)	74.28	**78.26**	73.91	77.90
	F1 (%)	79.77	82.04	79.43	**82.32**
D.TW	ACC (%)	**67.63**	62.59	64.03	**67.63**
	F1 (%)	39.86	34.81	36.71	**39.95**
M.A.G	ACC (%)	62.99	62.99	57.14	**63.64**
	F1 (%)	**47.05**	46.28	35.97	45.68
M.C	ACC (%)	75.60	79.04	64.26	**79.73**
	F1 (%)	80.65	**82.72**	75.24	81.50
M.TW	ACC (%)	59.74	**60.39**	57.14	**60.39**
	F1 (%)	38.82	41.33	31.82	**42.88**
Pha.C	ACC (%)	74.01	77.04	73.31	**80.30**
	F1 (%)	81.34	82.79	80.93	**85.79**
Pro.A.G	ACC (%)	85.37	85.37	84.39	**86.83**
	F1 (%)	76.76	**77.48**	72.64	69.78
Pro.C	ACC (%)	82.82	83.85	81.10	**85.22**
	F1 (%)	88.79	89.15	87.30	**90.62**
Pro.TW	ACC (%)	**82.44**	81.95	80.98	67.80
	F1 (%)	**42.96**	42.70	42.17	27.34

Table 3
The experimental results of semi-supervised learning for "75% label" and "50% label" cases.

Datasets	Label persentage	75%				50%			
	Metrics	TimeCLR (Traditional+DTW)	TimeCLR (DTW)	SimCLR	Supervised	TimeCLR (Traditional+DTW)	TimeCLR (DTW)	SimCLR	Supervised
D.A.G	ACC (%)	69.06	**70.50**	68.35	69.06	**68.35**	67.63	67.63	**68.35**
	F1 (%)	60.28	49.80	48.10	**67.51**	**48.03**	47.61	47.48	47.87
D.C	ACC (%)	75.72	**77.17**	74.28	75.72	74.64	**79.35**	73.55	73.19
	F1 (%)	81.44	**82.64**	80.97	79.38	80.23	**83.28**	78.47	78.11
D.TW	ACC (%)	66.19	58.27	57.55	**66.91**	**66.91**	58.27	57.55	65.47
	F1 (%)	38.27	25.65	25.30	**39.29**	**38.88**	25.65	24.87	38.18
M.A.G	ACC (%)	60.39	**62.34**	59.09	57.14	61.69	**63.64**	59.09	62.34
	F1 (%)	**40.37**	39.49	39.53	24.24	37.93	**42.38**	38.29	39.49
M.C	ACC (%)	73.88	78.69	67.01	**79.38**	73.88	**74.91**	67.01	**74.91**
	F1 (%)	79.46	82.49	76.12	**83.78**	79.35	79.78	76.24	**81.42**
M.TW	ACC (%)	**60.39**	57.79	57.79	**60.39**	57.79	**59.09**	56.49	57.79
	F1 (%)	34.88	**38.53**	32.54	34.41	**32.77**	32.16	32.16	29.56
Pha.C	ACC (%)	72.26	75.52	71.68	**80.54**	71.56	72.61	68.88	**73.66**
	F1 (%)	80.20	82.35	80.03	**85.81**	80.48	80.94	78.10	**81.92**
Pro.A.G	ACC (%)	**85.37**	84.39	83.41	**85.37**	84.88	**85.85**	82.93	79.02
	F1 (%)	**75.25**	74.58	58.26	59.32	64.27	**68.46**	68.38	60.78
Pro.C	ACC (%)	83.85	82.82	81.44	**84.54**	**83.16**	79.04	78.69	76.98
	F1 (%)	89.04	88.64	87.56	**89.84**	**88.89**	86.53	85.78	81.94
Pro.TW	ACC (%)	81.95	**82.93**	80.00	67.80	**81.95**	67.80	79.02	67.80
	F1 (%)	42.79	**43.31**	41.74	27.34	**42.87**	27.34	41.32	27.34

5.4. Transfer learning experiments

We build six transfer learning tasks using datasets that belong to the same classification task but do not belong to the same domain: *Pro.A.G* $\Rightarrow$ *D.A.G*, *Pro.C* $\Rightarrow$ *D.C*, *Pro.TW* $\Rightarrow$ *D.TW*, *Pro.A.G* $\Rightarrow$ *M.A.G*, *Pro.C* $\Rightarrow$ *M.C* and *Pro.TW* $\Rightarrow$ *M.TW*. The dataset before "$\Rightarrow$" is the source dataset, which we use for self-supervised training of the autoencoder used in the DTW data augmentation and self-supervised contrastive learning. The dataset after "$\Rightarrow$" is the target dataset we use for linear evaluation. For supervised learning, we pre-train the InceptionTime on the source dataset and then transfer it to the target dataset for fine-tuning.

Table 5 shows the experimental results. It can be found that our proposed TimeCLR beats all baseline methods. TimeCLR outperformed the supervised case on 4 out of 6 tasks.

5.5. Cluster experiments

To compare the clustering performance of representation vectors, we conduct K-means clustering experiments for different methods on the representation space. For the supervised model, we choose the input vector of the last layer as the representation for clustering experiments. In addition, we also perform K-means clustering based on Euclidean distance and K-means clustering based on DTW and DTW Barycenter Averaging [46] on the original time series data. We set the number of clusters to the number of classes of the dataset by default.

We choose Davies–Bouldin score [47] and Silhouette Coefficient [48] as the metrics. The Davies–Bouldin score is defined as the average similarity of each cluster to its most similar cluster, where similarity is the ratio of intra-cluster distance to inter-cluster distance. Clusters that are farther apart and have less dispersion will result in better scores, and smaller scores indicate better clustering. The Silhouette Coefficient measures how similar each sample is to its cluster compared to other clusters. The best value is 1 and the worst value is −1.

The experimental results are shown in Table 6. Experimental results show that the clustering performance of the representations generated by our framework is comparable to the

Table 4
The experimental results of semi-supervised learning for "25% label" and "10% label" cases.

Datasets	Label Persentage	25%				10%			
	Metrics	TimeCLR (Traditional+DTW)	TimeCLR (DTW)	SimCLR	Supervised	TimeCLR (Traditional+DTW)	TimeCLR (DTW)	SimCLR	Supervised
D.A.G	ACC (%)	63.31	**66.19**	60.43	54.68	66.91	**67.63**	63.31	50.36
	F1 (%)	44.47	**46.51**	42.57	33.45	46.86	47.47	**53.92**	26.98
D.C	ACC (%)	74.28	**76.09**	72.83	72.46	70.65	**74.64**	68.12	66.67
	F1 (%)	80.11	**81.25**	78.01	79.23	76.79	**78.40**	75.28	71.95
D.TW	ACC (%)	**62.59**	58.27	55.40	58.27	**61.87**	58.27	56.12	58.27
	F1 (%)	**34.36**	25.76	23.61	25.76	**35.00**	25.46	28.91	25.21
M.A.G	ACC (%)	**61.69**	**61.69**	59.09	49.35	24.68	**29.22**	18.83	24.68
	F1 (%)	37.93	**39.82**	38.29	31.05	23.87	**26.17**	10.56	15.20
M.C	ACC (%)	**65.64**	63.92	62.89	52.92	53.26	**57.04**	52.23	47.08
	F1 (%)	72.07	**72.87**	68.60	35.07	66.83	**72.65**	65.85	18.95
M.TW	ACC (%)	50.65	**51.30**	47.40	46.75	**55.84**	**55.84**	55.19	40.26
	F1 (%)	31.48	**31.85**	28.80	29.70	25.44	**25.80**	25.00	16.74
Pha.C	ACC (%)	**68.53**	68.30	66.43	63.64	66.55	**67.83**	65.62	51.05
	F1 (%)	**79.04**	77.03	77.71	77.06	77.63	**78.30**	77.43	41.50
Pro.A.G	ACC (%)	**84.39**	83.90	80.00	72.20	82.93	**83.90**	79.51	60.98
	F1 (%)	**84.39**	75.29	67.62	64.10	68.38	**73.02**	64.08	51.40
Pro.C	ACC (%)	**80.76**	80.07	79.04	72.51	**80.41**	**80.41**	76.63	54.98
	F1 (%)	**86.60**	86.12	85.91	83.26	**87.02**	86.59	84.33	50.94
Pro.TW	ACC (%)	79.51	**80.00**	75.61	67.80	5.37	**5.85**	5.37	**5.85**
	F1 (%)	41.06	**41.28**	38.41	27.34	3.71	**4.01**	3.71	**4.01**

Table 5
The experimental results of transfer learning tasks. The dataset before "⇒" is the source dataset we use for all self-supervised learning in TimeCLR, and the dataset after "⇒" is the target dataset we use for linear evaluation.

Tasks	Metrics	TimeCLR (Traditional+DTW)	TimeCLR (DTW)	SimCLR	Supervised
Pro.A.G⇒D.A.G	ACC (%)	74.10	**74.82**	73.38	74.10
	F1 (%)	74.41	**75.97**	70.69	73.30
Pro.C⇒D.C	ACC (%)	**76.81**	76.45	74.64	74.28
	F1 (%)	**81.40**	81.38	79.89	80.55
Pro.TW⇒D.TW	ACC (%)	**68.35**	67.63	64.75	66.91
	F1 (%)	**40.63**	39.95	37.20	39.52
Pro.A.G⇒M.A.G	ACC (%)	62.34	63.64	58.44	**64.29**
	F1 (%)	39.49	45.73	35.27	**48.85**
Pro.C⇒M.C	ACC (%)	75.60	74.57	74.23	**78.35**
	F1 (%)	80.55	79.44	80.11	**83.72**
Pro.TW⇒M.TW	ACC (%)	**58.44**	57.79	55.19	56.49
	F1 (%)	33.63	**39.63**	34.87	27.15

supervised model. Although the supervised model can produce better clustering performance since the input of the last layer of the supervised model will be performing classification, our framework is still able to produce representations with clustering performance close to the supervised model. In addition, we find that traditional data augmentations can corrupt the clustering results of the representations.

5.6. Ablation study

To study the effectiveness of the components of TimeCLR, we construct two series of experiments.

The first series of experiments are designed to study the DTW data augmentation. We set up four scenarios for this series of experiments:

- **SimCLR(Combine)** As shown in Fig. 7, the combination of Jitter and TimeWarp can produce surface-level effects similar to the DTW data augmentation, we replace the DTW data augmentation in "TimeCLR(DTW)" described in Section 5.2 with the combination of Jitter and TimeWarp in this scenario.
- **SimCLR(Traditional+Combine)** Based on the "SimCLR(Combine)" scenario, we add the six traditional time series data augmentations in this scenario.
- **TimeCLR(Random Encoder)** In this scenario, to investigate the utility of the special self-supervised learning process for DTW data augmentation, we use a naive autoencoder to replace the autoencoder used in DTW data augmentation. This naive autoencoder differs from the autoencoder used in the DTW data augmentation only in the training method, its training process is shown in Fig. 8. Since the parameters of the encoder block are randomly assigned before training, the output space of the encoder block is also random. We replace the DTW data augmentation in "TimeCLR(DTW)" with the data augmentation using a random encoder block.
- **TimeCLR(Traditional+Random Encoder)** Based on the "TimeCLR(Random Encoder)" scenario, we add the six traditional time series data augmentations in this scenario.

The experimental results are shown in Table 7. We observe that our proposed TimeCLR wins all scenarios. This series of experiments demonstrates the importance of DTW data augmentation for TimeCLR and the importance of our proposed special self-supervised training process for DTW data augmentation.

The second series of experiments are designed to study the contrastive learning framework. The difference between TimeCLR and traditional SimCLR which is used in the field of computer vision is not only that they use different data augmentation methods, but also that TimeCLR use a feature extractor based on InceptionTime, thus we set up three scenarios for this series of experiments:

- **ResNet** In this scenario, we replace the feature extractor in the self-supervised contrastive learning framework with the ResNet proposed by [14].

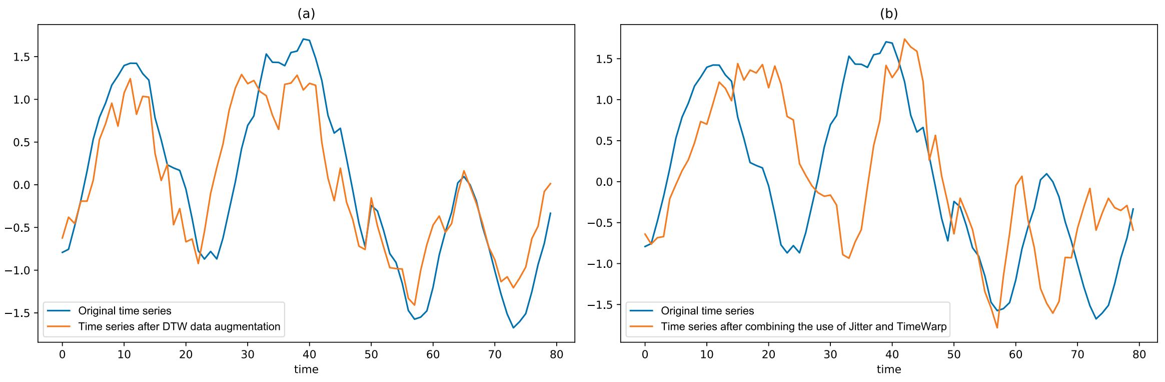

Fig. 7. Comparison of DTW data augmentation and the combination of Jitter and TimeWarp. The original time series is selected from the *D.A.G* dataset. (a) Comparison of the time series after DTW data augmentation operation and the original time series; (b) Comparison of the time series after combining the use of Jitter and TimeWarp and the original time series.

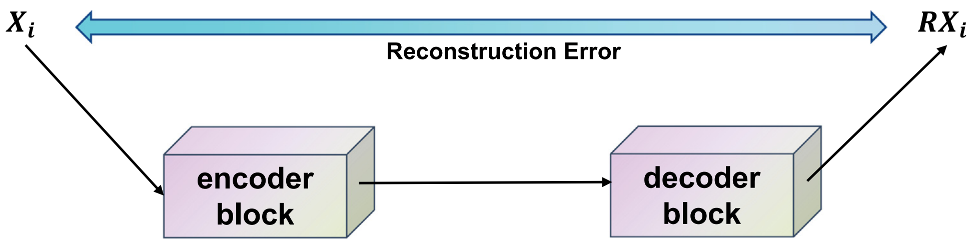

Fig. 8. The training process of the naive autoencoder. We only minimize the reconstruction error during training. The architecture of the naive autoencoder is exactly the same as the encoder block and the decoder block used in the DTW data augmentation.

Table 6
The experimental results of Cluster experiments. "DBS" is short for Davies–Bouldin score. "K-means(Euclidean)" represents perform K-means clustering based on Euclidean distance on the original time series data."K-means(DTW)" represents perform K-means clustering based on DTW and DTW Barycenter Averaging on the original time series data.

Datasets	Metrics	TimeCLR (Traditional+DTW)	TimeCLR (DTW)	SimCLR	Supervised	K-means (Euclidean)	K-means (DTW)
D.A.G	DBS↓	1.10	0.83	1.54	**0.82**	1.33	1.71
	Silhouette↑	0.35	0.49	0.22	**0.50**	0.30	0.32
D.C	DBS↓	1.34	1.34	1.82	1.12	1.39	**0.96**
	Silhouette↑	0.29	0.29	0.22	0.28	0.23	**0.39**
D.TW	DBS↓	1.34	1.34	1.41	**1.03**	1.46	2.59
	Silhouette↑	0.18	**0.33**	0.21	**0.33**	0.32	0.23
M.A.G	DBS↓	1.39	1.25	1.59	**1.00**	1.33	1.03
	Silhouette↑	0.27	**0.36**	0.24	0.32	0.29	0.31
M.C	DBS↓	1.07	0.96	1.03	**0.95**	1.15	1.67
	Silhouette↑	0.31	**0.50**	0.36	0.33	0.43	0.16
M.TW	DBS↓	1.29	**0.76**	1.39	1.04	1.36	1.80
	Silhouette↑	0.32	**0.54**	0.27	0.29	0.21	0.23
Pha.C	DBS↓	1.27	1.13	1.21	**1.11**	1.26	1.37
	Silhouette↑	0.25	0.26	0.25	**0.29**	0.23	0.28
Pro.A.G	DBS↓	1.18	1.35	1.35	**1.01**	1.63	2.43
	Silhouette↑	**0.28**	0.25	0.24	0.27	0.16	0.33
Pro.C	DBS↓	1.65	1.19	1.61	1.19	1.35	**0.78**
	Silhouette↑	0.22	0.22	0.25	0.35	0.27	**0.52**
Pro.TW	DBS↓	1.22	1.29	1.44	**1.12**	1.47	3.52
	Silhouette↑	0.23	**0.33**	0.23	0.31	0.31	0.02

- **3-layer MLP** In this scenario, we replace the feature extractor in the self-supervised contrastive learning framework with a 3-layer MLP. The architecture of 3-layer MLP is shown in Fig. 9.
- **Random InceptionTime** To investigate the effectiveness of the self-supervised contrastive learning framework, we perform the linear evaluation directly in this scenario using InceptionTime with randomly assigned parameters.

The experimental results are shown in Table 8, where we observe that TimeCLR still outperforms the other scenarios. The "Random InceptionTime" scenario performs the worst, which illustrates the importance of a self-supervised contrastive learning framework for obtaining high-quality representations. Secondly, we find that when the feature extractor model is larger, it yields better results. Specifically, 3-layer MLP produces the worst results, ResNet produces medium results, and InceptionTime produces the best results. This result indicates that using InceptionTime is most beneficial for extracting structure and feature information from time series, resulting in high-quality representations.

Table 7
The experimental results of the first series ablation experiments.

Datasets	Metrics	TimeCLR (Traditional +DTW)	TimeCLR (DTW)	SimCLR (Combine)	SimCLR (Traditional +Combine)	TimeCLR (Random Encoder)	TimeCLR (Traditional +Random Encoder)
D.A.G	ACC (%)	**74.10**	72.66	71.94	70.50	68.35	71.94
	F1 (%)	**73.88**	70.75	68.61	65.94	47.97	67.03
D.C	ACC (%)	74.28	**78.26**	73.55	73.91	58.33	73.91
	F1 (%)	79.77	**82.04**	79.55	79.55	73.68	79.78
D.TW	ACC (%)	**67.63**	62.59	61.87	58.27	58.27	58.27
	F1 (%)	**39.86**	34.81	34.31	25.46	25.65	25.37
M.A.G	ACC (%)	**62.99**	**62.99**	59.74	58.44	59.74	61.04
	F1 (%)	47.05	46.28	**50.60**	49.28	36.39	45.28
M.C	ACC (%)	75.60	**79.04**	73.88	71.13	63.92	63.57
	F1 (%)	80.65	**82.72**	78.89	76.14	73.15	75.46
M.TW	ACC (%)	59.74	**60.39**	59.09	59.09	57.14	53.25
	F1 (%)	38.82	**41.33**	34.12	36.99	32.79	33.36
Pha.C	ACC (%)	74.01	**77.04**	70.05	70.28	70.86	70.75
	F1 (%)	81.34	**82.79**	79.68	78.77	78.41	78.64
Pro.A.G	ACC (%)	**85.37**	**85.37**	82.93	83.41	84.88	83.90
	F1 (%)	76.76	**77.48**	74.81	71.38	74.34	75.44
Pro.C	ACC (%)	82.82	**83.85**	81.79	76.98	76.63	73.20
	F1 (%)	88.79	**89.15**	87.87	84.81	84.40	82.27
Pro.TW	ACC (%)	**82.44**	81.95	79.51	80.98	79.51	80.98
	F1 (%)	**42.96**	42.70	41.47	42.29	41.05	42.29

Table 8
The experimental results of the second series ablation experiments.

Datasets	Feature extractor	InceptionTime		ResNet		3-layer MLP		Random
	Metrics	TimeCLR (Traditional +DTW)	TimeCLR (DTW)	TimeCLR (Traditional +DTW)	TimeCLR (DTW)	TimeCLR (Traditional +DTW)	TimeCLR (DTW)	InceptionTime
D.A.G	ACC (%)	**74.10**	72.66	67.63	67.63	67.63	66.91	46.76
	F1 (%)	**73.88**	70.75	47.42	47.71	47.31	46.78	21.24
D.C	ACC (%)	74.28	**78.26**	74.28	76.81	70.29	69.93	58.33
	F1 (%)	79.77	**82.04**	79.18	81.29	78.07	78.44	73.68
D.TW	ACC (%)	**67.63**	62.59	64.75	64.75	58.27	58.27	30.22
	F1 (%)	**39.86**	34.81	36.50	36.65	25.29	25.46	7.73
M.A.G	ACC (%)	**62.99**	**62.99**	61.69	61.04	60.39	61.69	18.83
	F1 (%)	**47.05**	46.28	37.93	39.27	37.93	37.93	10.56
M.C	ACC (%)	75.60	**79.04**	71.13	76.29	60.48	60.14	57.04
	F1 (%)	80.65	**82.72**	78.46	81.10	74.04	73.99	72.65
M.TW	ACC (%)	59.74	**60.39**	56.49	59.09	55.84	55.84	27.27
	F1 (%)	38.82	**41.33**	31.37	38.65	29.79	25.68	7.14
Pha.C	ACC (%)	74.01	**77.04**	68.41	72.26	67.37	67.13	61.31
	F1 (%)	81.34	**82.79**	77.99	80.78	78.56	78.17	76.01
Pro.A.G	ACC (%)	**85.37**	**85.37**	84.39	84.39	84.88	84.39	48.78
	F1 (%)	76.76	**77.48**	70.12	75.01	58.96	59.33	21.86
Pro.C	ACC (%)	82.82	**83.85**	81.79	81.79	76.98	75.95	68.38
	F1 (%)	88.79	**89.15**	87.87	88.20	85.14	84.98	81.22
Pro.TW	ACC (%)	**82.44**	81.95	**82.44**	80.98	74.63	67.80	35.12
	F1 (%)	**42.96**	42.70	42.88	42.00	37.36	27.34	8.66

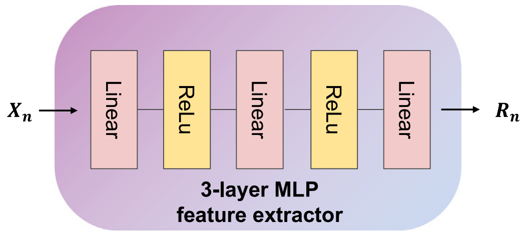

Fig. 9. The architecture of the 3-layer MLP used in the ablation study.

5.7. Hyperparameter sensitivity

We perform hyperparameter sensitivity experiments on the *D.A.G* dataset to study two critical hyperparameters in the TimeCLR: the standard deviation S_{noise} of the normal distribution that generates the noise vector $\boldsymbol{N}$ in the DTW data augmentation and the temperature τ of Eq. (15).

We first fix $\tau = 2.0$ and tune the standard deviation $S_{noise} \in \{0.5, 1.0, 2.0, 4.0, 8.0\}$. Figs. 10(a) and 10(b) show the effect of S_{noise}. We find that the two cases described in Section 5.2 achieve better results when $S_{noise} = 4.0$. Then we fix $S_{noise} = 4.0$ and tune the temperature $\tau \in \{0.05, 0.1, 0.2, 0.4, 0.8\}$. Figs. 10(c) and 10(d) show the effect of τ. We observed that all cases yielded the best performance when $\tau = 0.2$. In addition, we also find that the metric F1-score is more sensitive to hyperparameter settings. Based on the results in this subsection, we set S_{noise} to 4 and set τ to 0.2 in all experiments.

5.8. Apply TimeCLR to other datasets

To demonstrate the generality of the application domain of TimeCLR, we conducted experiments on 22 time series classification datasets[1] belonging to multiple domains. The details of these datasets are shown in Table 9.

[1] These datasets are available at http://www.timeseriesclassification.com/.

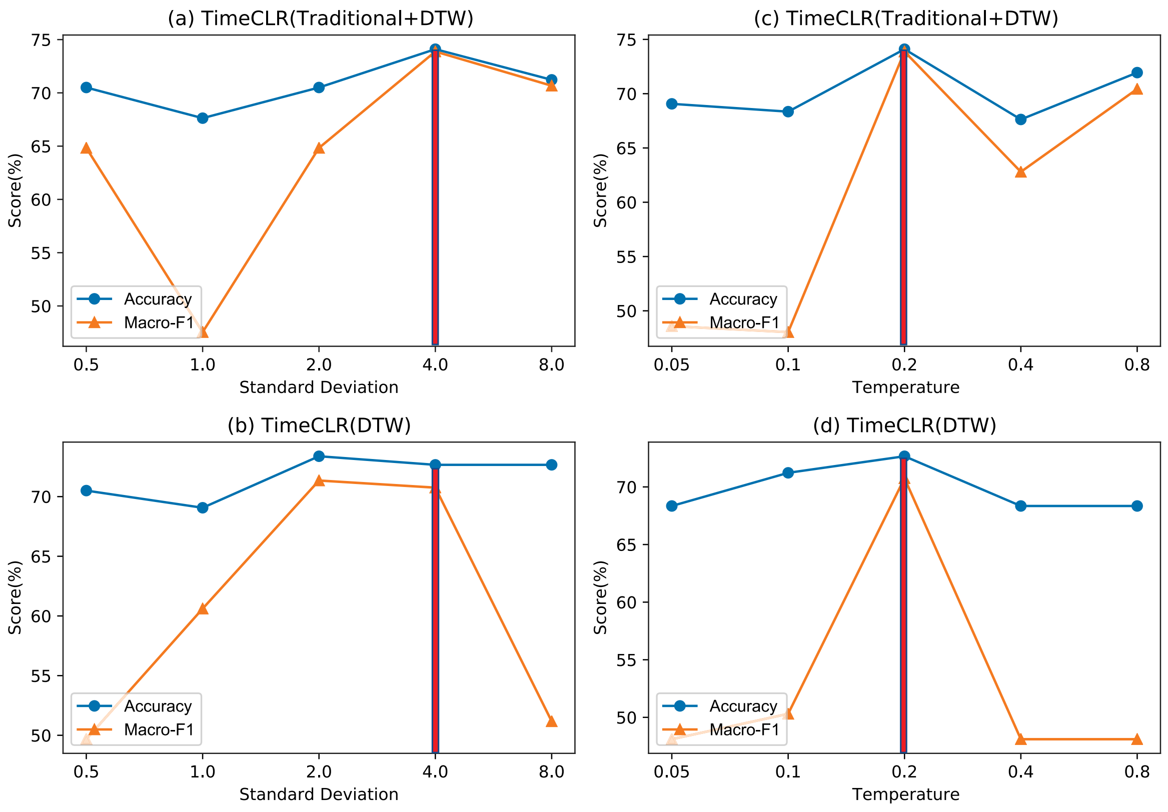

Fig. 10. The sensitivity analysis on *D.A.G* dataset. We first fix the temperature τ and tune the standard deviation S_{noise} for the following two cases. (a) The data augmentation used in TimeCLR contains both DTW data augmentation and traditional time series data augmentation; (b) The data augmentation used in TimeCLR is only the DTW data augmentation. Then we fix S_{noise} and tune τ for the following two cases. (c) The data augmentation used in TimeCLR contains both DTW data augmentation and traditional time series data augmentation; (d) The data augmentation used in TimeCLR is only the DTW data augmentation. The red vertical lines in the figure represent the values we have chosen.

Table 9
Details of time series datasets used in experiments on multiple domains.

Datasets	Number of classes	Size of training set	Size of testing set	Length	Domains
Beef	5	30	30	470	SPECTRO
BeetleFly	2	20	20	512	IMAGE
BirdChicken	2	20	20	512	IMAGE
Coffee	2	28	28	286	SPECTRO
DiatomSizeReduction	4	16	306	345	IMAGE
ECG200	2	100	100	96	ECG
FaceFour	4	24	88	350	IMAGE
GunPoint	2	50	150	150	MOTION
Ham	2	109	105	431	SPECTRO
Herring	2	64	64	512	IMAGE
ItalyPowerDemand	2	67	1029	24	SENSOR
Lighting2	2	60	61	637	SENSOR
Lighting7	7	70	73	319	SENSOR
Meat	3	60	60	448	SPECTRO
OliveOil	4	30	30	570	SPECTRO
Plane	7	105	105	144	SENSOR
ShapeletSim	2	20	180	500	SIMULATED
SharePriceIncrease	2	965	965	60	FINANCIAL
SonyAIBORobotSurf.1	2	20	601	70	SENSOR
ToeSegmentation2	2	36	130	343	MOTION
Trace	4	100	100	275	SENSOR
WormsTwoClass	2	181	77	900	MOTION

These datasets also have been divided into the training set and the testing set. We perform experiments using TimeCLR in two cases described in Section 5.2 and using the two baselines described in Section 5.1.3. The experimental results are shown in Table 10. Experimental results show that our method produces representations of significantly better quality than SimCLR. On these 22 datasets, our proposed TimeCLR performs slightly better than the supervised InceptionTime. The experimental results once again demonstrate that our proposed TimeCLR with DTW data augmentation is more beneficial for learning high-quality time series representations. Also, the experimental results show that TimeCLR can be applied to time series data representation in various domains.

Finally, we present the additional costs of TimeCLR on these 22 datasets in Table 11. The Pre-training and storage of the autoencoder for DTW data augmentation in TimeCLR leads to

Table 10
The experimental results on datasets from multiple domains.

Datasets	Metrics	TimeCLR (Traditional+DTW)	TimeCLR (DTW)	SimCLR	Supervised
Beef	ACC (%)	**63.33**	60.00	**63.33**	**63.33**
	F1 (%)	**63.77**	60.00	62.82	57.63
BeetleFly	ACC (%)	**95.00**	90.00	85.00	90.00
	F1 (%)	**94.74**	88.89	82.35	88.89
BirdChicken	ACC (%)	**100.00**	**100.00**	95.00	**100.00**
	F1 (%)	**100.00**	**100.00**	95.24	**100.00**
Coffee	ACC (%)	**100.00**	**100.00**	96.43	**100.00**
	F1 (%)	**100.00**	**100.00**	96.00	**100.00**
DiatomSizeReduction	ACC (%)	**94.12**	89.22	91.50	83.99
	F1 (%)	**93.07**	71.10	91.39	66.94
ECG200	ACC (%)	83.00	**91.00**	87.00	90.00
	F1 (%)	87.41	**93.02**	90.08	92.42
FaceFour	ACC (%)	**88.64**	87.50	86.36	62.50
	F1 (%)	**88.50**	87.49	86.17	54.93
GunPoint	ACC (%)	96.67	**99.33**	98.67	**99.33**
	F1 (%)	96.64	**99.33**	98.65	99.32
Ham	ACC (%)	75.24	**76.19**	72.38	71.43
	F1 (%)	**76.79**	75.25	71.84	69.39
Herring	ACC (%)	59.38	**64.06**	62.50	56.25
	F1 (%)	38.10	**51.06**	47.83	36.36
ItalyPowerDemand	ACC (%)	**95.43**	95.14	90.18	95.14
	F1 (%)	**95.37**	95.18	90.13	95.14
Lighting2	ACC (%)	67.21	**68.85**	67.21	54.10
	F1 (%)	**75.00**	74.67	73.68	70.21
Lighting7	ACC (%)	61.64	**64.38**	57.53	52.05
	F1 (%)	**54.44**	52.18	47.63	36.84
Meat	ACC (%)	**88.33**	**88.33**	**88.33**	**88.33**
	F1 (%)	**88.43**	88.15	87.96	88.30
OliveOil	ACC (%)	**70.00**	**70.00**	63.33	53.33
	F1 (%)	**60.41**	**60.41**	55.37	30.51
Plane	ACC (%)	**100.00**	**100.00**	99.05	**100.00**
	F1 (%)	**100.00**	**100.00**	99.08	**100.00**
ShapeletSim	ACC (%)	**100.00**	**100.00**	**100.00**	99.44
	F1 (%)	**100.00**	**100.00**	**100.00**	99.44
SharePriceIncrease	ACC (%)	**68.32**	68.22	67.08	67.81
	F1 (%)	23.12	**26.38**	22.82	23.96
SonyAIBORobotSurf.1	ACC (%)	93.84	92.85	91.18	**95.84**
	F1 (%)	93.26	92.28	90.59	**95.38**
ToeSegmentation2	ACC (%)	89.23	88.46	90.00	**90.00**
	F1 (%)	72.00	70.59	69.77	**72.34**
Trace	ACC (%)	98.00	**100.00**	98.00	**100.00**
	F1 (%)	97.78	**100.00**	97.78	**100.00**
WormsTwoClass	ACC (%)	**79.22**	76.62	77.92	57.14
	F1 (%)	**82.98**	81.25	82.11	72.73

Table 11
The additional cost of TimeCLR.

Datasets	Additional time cost (s)		Additional memory cost
	Generate training data	Pre-training	
Beef	0.19	9.70	143.2 MB
BeetleFly	0.08	5.04	169.9 MB
BirdChicken	0.05	5.13	169.9 MB
Coffee	0.07	4.89	53 MB
DiatomSizeReduction	0.02	2.10	77.2 MB
ECG200	0.48	60.09	5.7 MB
FaceFour	0.06	5.59	79.4 MB
GunPoint	0.16	15.69	14.6 MB
Ham	1.54	110.98	120.4 MB
Herring	0.52	44.85	169.9 MB
ItalyPowerDemand	0.10	26.27	381 KB
Lighting2	0.64	54.28	263 MB
Lighting7	0.46	34.93	66 MB
Meat	0.44	33.91	130.1 MB
OliveOil	0.13	12.21	210.6 MB
Plane	0.60	66.51	13.5 MB
ShapeletSim	0.07	4.85	162 MB
SharePriceIncrease	34.03	5591.10	2.3 MB
SonyAIBORobotSurf.1	0.02	2.68	3.2 MB
ToeSegmentation2	0.16	9.81	76.3 MB
Trace	0.86	64.74	49 MB
WormsTwoClass	7.76	789.63	525 MB

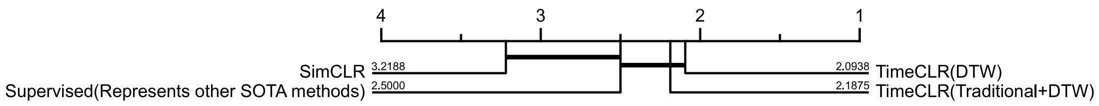

Fig. 11. Critical difference diagram illustrating the accuracy of different methods.

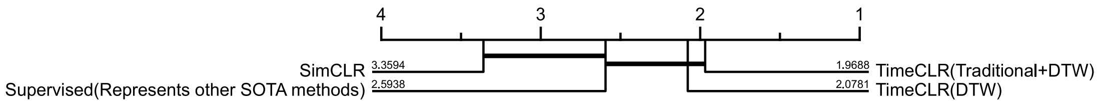

Fig. 12. Critical difference diagram illustrating the F1-score of different methods.

additional costs compared to other self-supervised representation learning methods. The additional time cost is primarily from the generation of training data and the pre-training of the autoencoder. The additional memory cost is from the storage of the *pkl* file of the autoencoder. As the table shows, the time and memory costs of TimeCLR are small on most of the datasets. The main reason for the slightly higher costs on a few datasets is that excessive training data are generated, which inspires us to reduce the amount of data required for autoencoder pre-training according to reality.

In addition, we generated critical difference diagrams of accuracy and F1-score based on experimental results on the 32 datasets used in this paper. As shown in Figs. 11 and 12, TimeCLR performs slightly better than supervised InceptionTime, which represents the other state-of-the-art self-supervised time series representation methods. After paying the additional time and memory costs, TimeCLR makes a significant difference from SimCLR, thus the additional cost is acceptable for TimeCLR.

6. Conclusion

In this paper, we propose a novel self-supervised contrastive learning framework for univariate time series representation learning named TimeCLR, which aims to use the structure and feature information of unlabeled time series to obtain high-quality representations through self-supervised learning. Inspired by the process of measuring time series similarity by DTW, we propose the DTW data augmentation, which uses a special self-supervised trained autoencoder that not only produces the phase shift and amplitude change phenomena targeted by DTW, but also retains the structure and feature information of the time series. We successfully extend SimCLR to time series representation field using the DTW data augmentation combined with the powerful feature extraction capability of the InceptionTime. Our TimeCLR framework enables the feature extractor to learn the structure and feature information of a time series by maximizing the similarity of the views obtained from the same time series after two separate data augmentation operations. Ultimately, the feature extractor is able to produce high-quality representations in an end-to-end manner. Multiple types of experiments validate the importance of each component of TimeCLR, not only showing that TimeCLR yields comparable performance to the supervised InceptionTime model, but also showing that TimeCLR can produce better performance when labeled time series is insufficient as well as can be used for univariate time series data in different domains. In the future, we will explore the possibility of applying the framework to multivariate time series and leveraging DTW to further improve the architecture of the feature extractor.

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.

Acknowledgment

This work is supported by the National Natural Science Foundation of China [grant number 62162062].

References

[1] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: A review, Data Min. Knowl. Discov. 33 (4) (2019) 917–963.
[2] B. Lim, S. Zohren, Time-series forecasting with deep learning: A survey, Phil. Trans. R. Soc. A 379 (2194) (2021) 20200209.
[3] T. Ching, D.S. Himmelstein, B.K. Beaulieu-Jones, A.A. Kalinin, B.T. Do, G.P. Way, E. Ferrero, P.-M. Agapow, M. Zietz, M.M. Hoffman, et al., Opportunities and obstacles for deep learning in biology and medicine, J. R. Soc. Interface 15 (141) (2018) 20170387.
[4] T. Chen, S. Kornblith, M. Norouzi, G. Hinton, A simple framework for contrastive learning of visual representations, in: International Conference on Machine Learning, PMLR, 2020, pp. 1597–1607.
[5] T. Chen, S. Kornblith, K. Swersky, M. Norouzi, G.E. Hinton, Big self-supervised models are strong semi-supervised learners, in: H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, H. Lin (Eds.), Advances in Neural Information Processing Systems, 33, Curran Associates, Inc., 2020, pp. 22243–22255, https://proceedings.neurips.cc/paper/2020/file/fcbc95ccdd551da181207c0c1400c655-Paper.pdf.
[6] M. Noroozi, P. Favaro, Unsupervised learning of visual representations by solving jigsaw puzzles, in: European Conference on Computer Vision, Springer, 2016, pp. 69–84.
[7] Z. Wu, Y. Xiong, S.X. Yu, D. Lin, Unsupervised feature learning via non-parametric instance discrimination, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 3733–3742.
[8] K. He, H. Fan, Y. Wu, S. Xie, R. Girshick, Momentum contrast for unsupervised visual representation learning, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 9729–9738.
[9] R.D. Hjelm, A. Fedorov, S. Lavoie-Marchildon, K. Grewal, P. Bachman, A. Trischler, Y. Bengio, Learning deep representations by mutual information estimation and maximization, 2018, arXiv preprint arXiv:1808.06670.
[10] M.N. Mohsenvand, M.R. Izadi, P. Maes, Contrastive representation learning for electroencephalogram classification, in: Machine Learning for Health, PMLR, 2020, pp. 238–253.
[11] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: KDD Workshop, vol. 10, Seattle, WA, USA, 1994, pp. 359–370.
[12] M. Müller, Dynamic time warping, Inf. Retr. Music Motion (2007) 69–84.
[13] Y.-S. Jeong, M.K. Jeong, O.A. Omitaomu, Weighted dynamic time warping for time series classification, Pattern Recognit. 44 (9) (2011) 2231–2240.
[14] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 International Joint Conference on Neural Networks, IJCNN, IEEE, 2017, pp. 1578–1585.
[15] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: A review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 31 (3) (2017) 606–660.
[16] H.I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D.F. Schmidt, J. Weber, G.I. Webb, L. Idoumghar, P.-A. Muller, F. Petitjean, Inceptiontime: Finding alexnet for time series classification, Data Min. Knowl. Discov. 34 (6) (2020) 1936–1962.
[17] L. Ye, E. Keogh, Time series shapelets: A new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 947–956.
[18] A. Mueen, E. Keogh, N. Young, Logical-shapelets: An expressive primitive for time series classification, in: Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2011, pp. 1154–1162.

[19] J. Hills, J. Lines, E. Baranauskas, J. Mapp, A. Bagnall, Classification of time series by shapelet transformation, Data Min. Knowl. Discov. 28 (4) (2014) 851–881.

[20] T. Górecki, M. Łuczak, Non-isometric transforms in time series classification using DTW, Knowl.-Based Syst. 61 (2014) 98–108.

[21] X. Cai, T. Xu, J. Yi, J. Huang, S. Rajasekaran, DTWNet: A dynamic time warping network, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper/2019/file/02f063c236c7eef66324b432b748d15d-Paper.pdf.

[22] F. Karim, S. Majumdar, H. Darabi, S. Chen, LSTM fully convolutional networks for time series classification, IEEE Access 6 (2017) 1662–1669.

[23] T.T. Um, F.M. Pfister, D. Pichler, S. Endo, M. Lang, S. Hirche, U. Fietzek, D. Kulić, Data augmentation of wearable sensor data for Parkinson's disease monitoring using convolutional neural networks, in: Proceedings of the 19th ACM International Conference on Multimodal Interaction, 2017, pp. 216–220.

[24] S. Haradal, H. Hayashi, S. Uchida, Biosignal data augmentation based on generative adversarial networks, in: 2018 40th Annual International Conference of the IEEE Engineering in Medicine and Biology Society, EMBC, IEEE, 2018, pp. 368–371.

[25] G. Ramponi, P. Protopapas, M. Brambilla, R. Janssen, T-cgan: Conditional generative adversarial network for data augmentation in noisy time series with irregular sampling, 2018, arXiv preprint arXiv:1811.08295.

[26] P. Cao, X. Li, K. Mao, F. Lu, G. Ning, L. Fang, Q. Pan, A novel data augmentation method to enhance deep neural networks for detection of atrial fibrillation, Biomed. Signal Process. Control 56 (2020) 101675.

[27] K. Kamycki, T. Kapuscinski, M. Oszust, Data augmentation with suboptimal warping for time-series classification, Sensors 20 (1) (2020) 98.

[28] T. DeVries, G.W. Taylor, Dataset augmentation in feature space, 2017, arXiv preprint arXiv:1702.05538.

[29] N. Komodakis, S. Gidaris, Unsupervised representation learning by predicting image rotations, in: International Conference on Learning Representations, ICLR, 2018.

[30] J. Wang, J. Jiao, Y.-H. Liu, Self-supervised video representation learning by pace prediction, in: European Conference on Computer Vision, Springer, 2020, pp. 504–521.

[31] A. Hyvarinen, H. Morioka, Unsupervised feature extraction by time-contrastive learning and nonlinear ica, Adv. Neural Inf. Process. Syst. 29 (2016) 3765–3773.

[32] J.-Y. Franceschi, A. Dieuleveut, M. Jaggi, Unsupervised scalable representation learning for multivariate time series, in: H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, R. Garnett (Eds.), Advances in Neural Information Processing Systems, Vol. 32, Curran Associates, Inc., 2019, https://proceedings.neurips.cc/paper/2019/file/53c6de78244e9f528eb3e1cda69699bb-Paper.pdf.

[33] S. Jawed, J. Grabocka, L. Schmidt-Thieme, Self-supervised learning for semi-supervised time series classification, Adv. Knowl. Discov. Data Min. 12084 (2020) 499.

[34] S. Tonekaboni, D. Eytan, A. Goldenberg, Unsupervised representation learning for time series with temporal neighborhood coding, in: International Conference on Learning Representations, 2021.

[35] H. Fan, F. Zhang, Y. Gao, Self-supervised time series representation learning by inter-intra relational reasoning, 2020, arXiv preprint arXiv:2011.13548.

[36] E. Eldele, M. Ragab, Z. Chen, M. Wu, C.K. Kwoh, X. Li, C. Guan, Time-series representation learning via temporal and contextual contrasting, in: Z.-H. Zhou (Ed.), Proceedings of the Thirtieth International Joint Conference on Artificial Intelligence, IJCAI-21, International Joint Conferences on Artificial Intelligence Organization, 2021, pp. 2352–2359, http://dx.doi.org/10.24963/ijcai.2021/324, Main Track.

[37] G. Anand, R. Nayak, Unsupervised visual time-series representation learning and clustering, in: International Conference on Neural Information Processing, Springer, 2020, pp. 832–840.

[38] A. Abid, J. Zou, Autowarp: Learning a warping distance from unlabeled time using sequence autoencoders, in: Proceedings of the 32nd International Conference on Neural Information Processing Systems, 2018, pp. 10568–10578.

[39] P. Sarkar, A. Etemad, Self-supervised ECG representation learning for emotion recognition, IEEE Trans. Affect. Comput. (2020).

[40] A. Mueen, E. Keogh, Extracting optimal performance from dynamic time warping, in: Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2016, pp. 2129–2130.

[41] K. Sohn, Improved deep metric learning with multi-class n-pair loss objective, Adv. Neural Inf. Process. Syst. (2016) 1857–1865.

[42] A.v.d. Oord, Y. Li, O. Vinyals, Representation learning with contrastive predictive coding, 2018, arXiv preprint arXiv:1807.03748.

[43] M.N. Mohsenvand, M.R. Izadi, P. Maes, Contrastive representation learning for electroencephalogram classification, in: Machine Learning for Health, PMLR, 2020, pp. 238–253.

[44] L.M. Davis, Predictive Modelling of Bone Ageing (Ph.D. thesis), University of East Anglia, 2013.

[45] F. Cao, H.K. Huang, E. Pietka, V. Gilsanz, Digital hand atlas and web-based bone age assessment: System design and implementation, Comput. Med. Imaging Graph. 24 (5) (2000) 297–307.

[46] F. Petitjean, A. Ketterlin, P. Gançarski, A global averaging method for dynamic time warping, with applications to clustering, Pattern Recognit. 44 (3) (2011) 678–693.

[47] D.L. Davies, D.W. Bouldin, A cluster separation measure, IEEE Trans. Pattern Anal. Mach. Intell. (2) (1979) 224–227.

[48] P.J. Rousseeuw, Silhouettes: A graphical aid to the interpretation and validation of cluster analysis, J. Comput. Appl. Math. 20 (1987) 53–65.