Wuhan University
Journal of Natural Sciences

2022, Vol.27 No.6, 521-530

Article ID 1007-1202(2022)06-0521-10
DOI https://doi.org/10.1051/wujns/2022276521

Self-Supervised Time Series Classification Based on LSTM and Contrastive Transformer

□ **ZOU Yuanhao, ZHANG Yufei, ZHAO Xiaodong†**

School of Electronic and Information Engineering, Tongji University, Shanghai 201804, China

Abstract: Time series data has attached extensive attention as multi-domain data, but it is difficult to analyze due to its high dimension and few labels. Self-supervised representation learning provides an effective way for processing such data. Considering the frequency domain features of the time series data itself and the contextual feature in the classification task, this paper proposes an unsupervised Long Short-Term Memory (LSTM) and contrastive transformer-based time series representation model using contrastive learning. Firstly, transforming data with frequency domain-based augmentation increases the ability to represent features in the frequency domain. Secondly, the encoder module with three layers of LSTM and convolution maps the augmented data to the latent space and calculates the temporal loss with a contrastive transformer module and contextual loss. Finally, after self-supervised training, the representation vector of the original data can be got from the pre-trained encoder. Our model achieves satisfied performances on Human Activity Recognition (HAR) and sleepEDF real-life datasets.

Key words: self-supervised learning; contrastive learning; time series classification

CLC number: TP 391

Received date: 2022-09-25

Foundation item: Supported by the National Key Research and Development Program of China (2019YFB1706401)

Biography: ZOU Yuanhao, male, Master candidate, research direction: cloud manufacturing resource combination. E-mail: 2133062@tongji.edu.cn

† To whom corresepondence should be addressed. E-mail: tjcad-zhaoxd@tongji.edu.cn

0 Introduction

In real life, the application of time series data covers various aspects such as finance, manufacturing, weather forecasting, and biomedicine[1]. To obtain feature in time series data, researchers in data mining have proposed a large number of algorithms, such as the Hidden Markov Model (HMM)[2], Linear Dynamical System (LDS)[3]. However, with the continuous expansion of the application field, people's requirements for tasks such as time series data classification and prediction are constantly increasing and traditional algorithms may not meet the requirements.

In recent years, people have tried to use deep learning methods to perform data mining on time series data. However, time series data have high dimensionality and high cost in directly processing the data, so the use of representation learning to reduce the dimension of time series data has become an important direction of time series data research[4]. Representation learning can reduce the dimensionality of time series data with few loss in features, thereby reducing the cost of processing the data[5]. Another major feature of time series is the lack of labels, and time series data often exist in a form that is not easy for people to learn and recognize, which makes it difficult for traditional supervised deep learning methods to be applied to actual time series data classification[6].

Self-supervised representation learning techniques have been widely used in the field of computer vision in recent years[7], which construct pseudo-label to help learn features. Doersch *et al*[8] proposed a model which

learned the hidden feature in photos by constructing a prediction task of different positions from the same photo. Gidaris *et al*[9] obtained the latent space representation of the image by feeding the rotation of the same image at different angles into the model. After that, contrastive learning[10] has become a hot direction in self-supervised learning. Contrastive learning artificially constructs labels by performing different augmentation on the same data to obtain multiple different data with the same hidden feature. Using the encoder module, the augmented data is mapped into latent space and the representation vector is got. Augmented data from the same original data are considered positive samples, and the rest of the samples are considered negative samples. Then loss function is used to maximize the similarity of vectors in the positive sample and minimize the similarity of vectors in the negative samples. In this way, the encoder can represent the original data well by learning the hidden feature.

In this paper, an improved model based on the time series representation learning framework via Temporal and Contextual Contrasting (TS-TCC) model[11] is proposed to classify time series data. Compared with the original model, this method can improve the model's ability to get the feature in the frequency domain of time series data and represent global feature in time series data. The model outperformed the original model on some test sets. Specifically, the innovations of this paper are as follows:

1) We adopt the Fast Fourier Transform (FFT) as the way of data augmentation, which improves the representation ability of the model for frequency domain features and increases the robustness of the model.

2) We design a multi-layer Long Short-Term Memory (LSTM) encoder module using composite convolution downsampling, which can obtain both local and global features on the augmented data.

3) We propose a contrastive transformer module for hidden feature extraction. This module increases the similarity of hidden features between augmented data in positive samples, thereby improving the representation ability of the model.

1 Related Work

Due to the excellent performance of self-supervised representation learning methods in the unsupervised domain and transfer learning, more and more scholars have applied self-supervised representation learning to time series data analysis in recent years. Sarkar *et al*[12] proposed a self-supervised learning model for electrocardiogram(ECG) data analysis by applying six different transformations to ECG data and using the transformation to create labels. By dividing the deep neural network into a set of gradient isolation modules, Löwe *et al*[13] used the InfoNCE loss to calculate the loss inside the module, and the greedy training method was used between the modules to maximally preserve the feature of its inputs. The Contrastive Predictive Coding (CPC) model proposed by Oord *et al*[14] uses the autoregressive model to predict future data with latent space data. By introducing probabilistic contrastive loss and negative sampling methods, the model has good results in speech, images, and other datasets. Franceschi *et al*[15] proposed a dilated causal convolution-based encoder and a triple loss function based on time-based negative sampling, which treated segments from different time series as pairs of negative samples and sub-segments from the same sample as positive sample pairs. This model was mainly used to classify time series data with inconsistent lengths. The Temporal Neighborhood Coding (TNC) model proposed by Tonekaboni *et al*[16] solved the problem of non-stationary time series classification. This model defined a time neighborhood. The samples in the neighborhood are considered positive samples, and the samples outside the domain are considered negative samples. To solve the problem that similar samples are located outside the time neighborhood caused by periodic time series, the author used positive-unlabeled learning to introduce a negative sample weight as a supplement to the loss function. The TS-TCC model proposed by Eldele *et al*[11] treated the data with two different augmentations as a pair of positive samples, mapped the augmented data to the latent space through the convolutional layer, and additionally learned the feature from latent space through the Transformer module. For further learning, the temporal loss function was calculated internally by the positive sample pair in a way similar to CPC[14]. The InfoNCE loss was used to calculate the loss between the positive and negative samples as a contextual loss.

The common problem of the above methods is that the time series data is only represented from the features in the time domain, but not from the perspective of the frequency domain. What's more, TS-TCC does not introduce more global features in the encoder stage, resulting

in fewer global features obtained in the latent space, which affects the effect of the model.

2 Methods

The model proposed in this paper is an improvement to the TS-TCC[11] model, and its architecture is shown in Fig. 1. The data is first augmented in two different ways, then mapped to the latent space through encoder to get a representation vector. The representation vectors from the same sample then calculate the temporal loss through the contrastive Transformer module. The result of the Transformer encoder is used as the contextual feature of the representation vector from the encoder. The contextual features from the same sample are considered positive samples, and the other features in the same batch are considered negative samples. The contextual loss is calculated by using the InfoNCE loss function with positive samples and negative samples. After learning for several epochs, the encoder module can transfer data into a representation vector with hidden features.

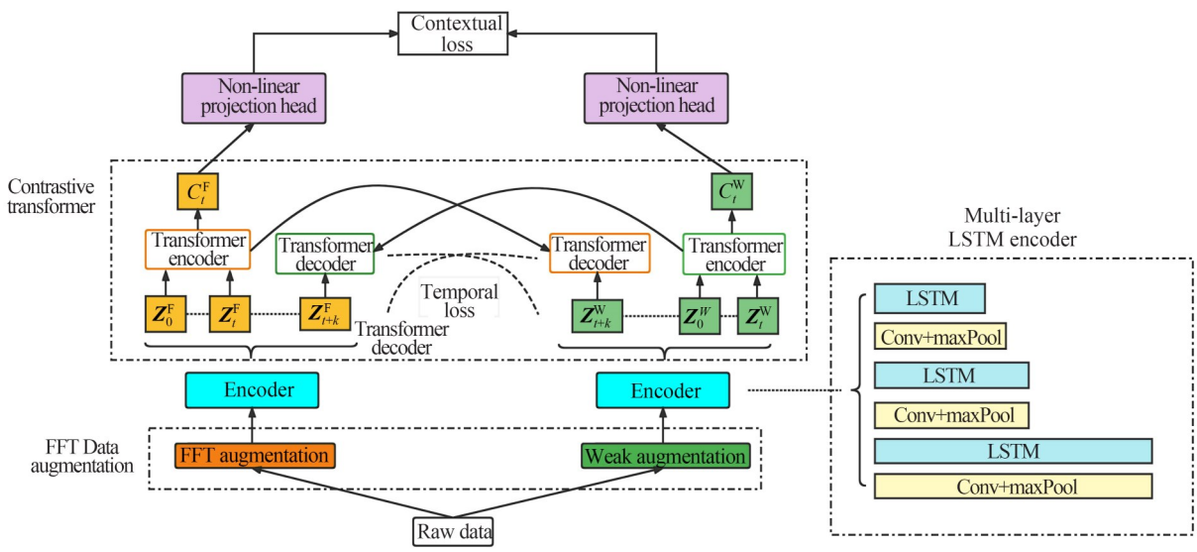


Fig. 1 Architecture of our model

2.1 FFT Data Augmentation

Data augmentation is a commonly used way to generate positive sample pairs in contrastive learning. However, at present, most of the enhancement methods for time series data use the augmentation method in the time domain and rarely in the frequency domain[17]. In order to obtain the both hidden features in the frequency domain and time domain, an augmentation method based on FFT is adopted in this paper. And in order to highlight the hidden feature, the paper also combines the FFT augmented data with weak augmented data into a positive sample pair to facilitate the comparison between them. This augmentation method also enhances the robustness of the model.

This paper performs two different augmentation operations on the original data $X \in \mathbb{R}^{\dim}$, $X = \{x_1, x_2, \cdots, x_n\}$. We denote X^{F} as the augmented sample with FFT augmentation, and X^{W} as the augmented sample with weak augmentation. The FFT augmentation method used in this paper is to first perform a permutation operation on the data, that is, randomly split data with maximum segments M and shuffle them, and then perform a FFT on the data to transfer it to the frequency domain. Scaling and warping operations are performed to the frequency domain data, that is, randomly warp and scale the data in the frequency domain, and the corresponding warping function follows a beta(α, α) distribution. Then the data is converted back to the time domain. For weak augmentation, only random Gaussian noise is added to it. Figure 2 shows the difference between augmentation in the time domain and the frequency domain. The scaling ratio is 2 and the parameter α is 0.5. The data is about the accelerometer of the volunteer.

For time series, the characteristics of the sequence change over time, and the change in frequency in some tasks is so closely related to the task that working in the frequency domain is more useful than that in the time domain[5]. The advantage of adding frequency domain aug-

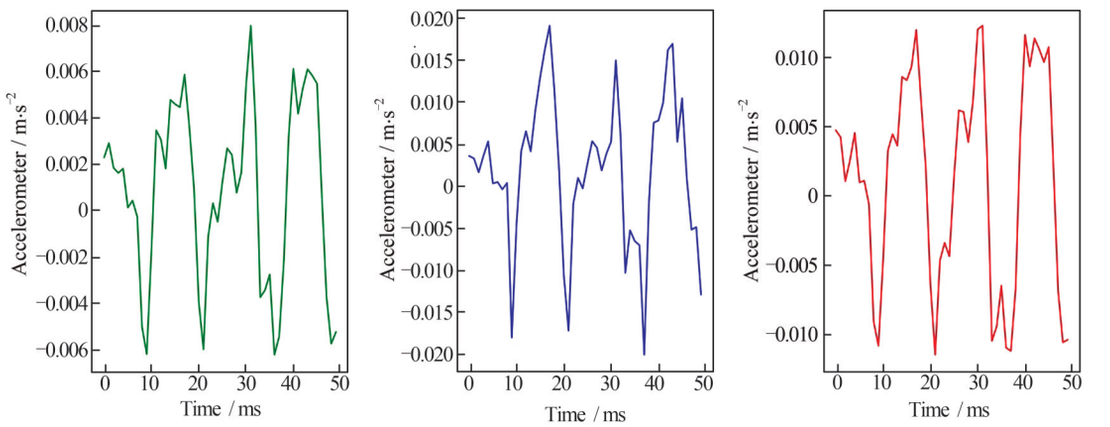


Fig. 2 Augmentation in time or frequency domain

mentation is that the feature in the frequency domain is considered without destroying the hidden features of the data, which is beneficial to further classify data with high-frequency similarity. As can be seen from Fig. 2, the augmentation based only on the time domain has serious feature loss in the two time periods of 10-20 ms and 40-50 ms, while the augmentation based on the frequency domain well preserves the change trend of the original data although the values are different.

2.2 Multi-Layer LSTM Encoder

The encoder module used in the TS-TCC[11] model is a 3-block convolution architecture, but the representation vector obtained by a simple 3-block convolution is difficult to contain the global feature of the time series. After referring to the downsampling method of Informer[18], this paper designs the encoder architecture as shown in Fig. 3. This module obtains local feature of time series data through convolution blocks and global feature through the LSTM module, and downsampling operation can reduce the processing time of data with less feature loss. Meanwhile, the LSTM module can well protect the time series logic of the data.

The function of the encoder module is to map the augmented high-dimensional time series data to the latent space $\mathbb{R}^d$ of dimensional d to obtain hidden vectors $\boldsymbol{Z}=f_{\text{encoder}}(X)$, $\boldsymbol{Z}\in\mathbb{R}^d$. Suppose $\boldsymbol{Z}=\{z_0, z_1, \cdots, z_T\}$, and its total length is T. Also suppose $\boldsymbol{Z}^{\text{F}}$ is the representation vector of FFT augmented data and $\boldsymbol{Z}^{\text{W}}$ is the representation vector of weak augmented data.

The time series data T in real life is generally large, and the 3-block convolution method can only collect its local feature, while the LSTM module can solve the problem of long-term data dependence and effectively improve the model's ability to obtain global feature. So

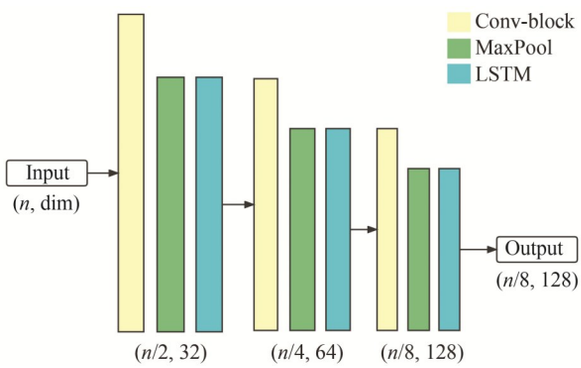


Fig. 3 Encoder with 3-layers LSTM

adding a single-layer LSTM to process the downsampling resulting from the convolution block can improve the mapping result of the encoder.

The output of the encoder is the representation vector of augmented data. It will also be used to transform the original data into representation vectors when making final classification predictions after self-supervised learning is complete. One Fully Connected (FC) layer is used to predict the class to which the representation vector belongs.

2.3 Contrastive Transformer

Transformer has been widely used in computer vision[19] and nature language processing fields[20] in recent years. It can effectively collect local and global feature in data, and use attention mechanisms to help predict future input. Therefore, Transformer shows good performance in translation[20], and this paper tries to use this property for time series data processing.

Compared with other models based on contrastive prediction such as CPC[14] and TS-TCC[11], our model utilizes the hidden feature of the predicted data segment to

a greater extent. Those models only use the contextual feature of the previous piece of data to predict the next piece of data, but our proposed contrast Transformer obtains both the contextual and local feature of the previous piece of feature. At the same time, the time continuity of the later piece of data is preserved in the prediction stage. Figure 4 shows the architecture of the contrastive Transformer module.

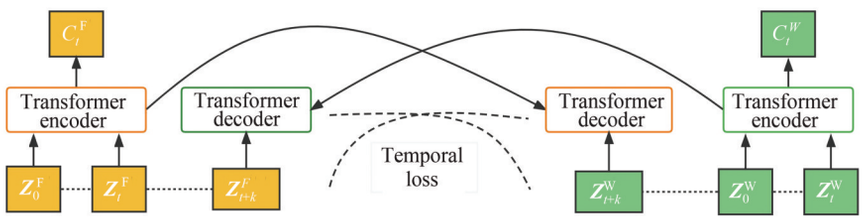


Fig. 4 Contrastive Transformer architecture

Like TS-TCC[11], this paper divides the time series into two parts: the first part is 0 to t items of the sequence $\{z_0, z_1, \cdots, z_t\}$, and the second part is t+1 to $t+k$ items of the sequence $\boldsymbol{Z}_{\text{output}} = \{z_{t+1}, z_{t+2}, \cdots, z_{t+k}\}$. Suppose c^t is the contextual vector and add it to the first part, so the first part $\boldsymbol{Z}_{\text{input}} = \{c^t, z_0, z_1, \cdots, z_t\}$. In this way, when the transformer encoder is learning, c^t can learn the attention relationship between itself and other elements in $\boldsymbol{Z}_{\text{input}}$, and these relationships can represent the context feature of z. However, TS-TCC[11] uses c^t to predict data in $\boldsymbol{Z}_{\text{output}}$ like the CPC[14] model, which hardly takes use of the hidden features in the $\boldsymbol{Z}_{\text{output}}$.

This paper proposes a Contrastive Transformer module. Firstly, the representation vector $\boldsymbol{Z}_{\text{input}}$ is used as the input of the Transformer encoder $f_{\text{TransEncoder}}(\cdot)$ to get contextual vector c^t and hidden feature $\boldsymbol{Z}_{\text{hidden}}$ as follows:

$$\begin{cases} \boldsymbol{Z}^{\text{W}}_{\text{hidden}} = f_{\text{TransEncoder}}(\boldsymbol{Z}^{\text{W}}_{\text{input}}) \\ c^{\text{W}}_t = \boldsymbol{Z}^{\text{W}}_{\text{hidden}}[0] \\ \boldsymbol{Z}^{\text{F}}_{\text{hidden}} = f_{\text{TransEncoder}}(\boldsymbol{Z}^{\text{F}}_{\text{input}}) \\ c^{\text{F}}_t = \boldsymbol{Z}^{\text{F}}_{\text{hidden}}[0] \end{cases} \tag{1}$$

Since $\boldsymbol{Z}^{\text{F}}$ and $\boldsymbol{Z}^{\text{W}}$ are essentially the results of two different augmented data maps from the same sample X, they have similar hidden features. Suppose $\boldsymbol{Z}^{\text{W}}{}_{\text{hidden}}$ is the hidden feature get by the Transformer encoder and $\boldsymbol{Z}^{\text{F}}{}_{\text{output}}$ is the vector that needs to be translated. Take both of them into the Transformer decoder, $f_{\text{TransDecoder}}(\cdot)$, and get the result $\boldsymbol{P}^{\text{W}}$. In the same way, we can get $\boldsymbol{P}^{\text{F}}$:

$$\begin{cases} \boldsymbol{P}^{\text{F}} = f_{\text{TransDecoder}}(\boldsymbol{Z}^{\text{F}}_{\text{hidden}}, \boldsymbol{Z}^{\text{W}}_{\text{output}}) \\ \boldsymbol{P}^{\text{W}} = f_{\text{TransDecoder}}(\boldsymbol{Z}^{\text{W}}_{\text{hidden}}, \boldsymbol{Z}^{\text{F}}_{\text{output}}) \end{cases} \tag{2}$$

If the representation vector learned well in encoder, $\boldsymbol{P}^{\text{W}}$ and $\boldsymbol{P}^{\text{F}}$ should have the same hidden feature as $\boldsymbol{Z}^{\text{W}}{}_{\text{hidden}}$ and $\boldsymbol{Z}^{\text{F}}{}_{\text{hidden}}$, respectively. So the temporal loss function L_{T} can be constructed.

$$\begin{cases} L^{\text{F}}_{\text{T}} = -\dfrac{1}{K}\displaystyle\sum_{k=1}^{K} \log \dfrac{\exp((P^{\text{F}}_{t+k})^{\text{T}} z^{\text{F}}_{t+k})}{\sum_{n\in[t,k]} \exp((P^{\text{F}}_{t+k})^{T} z^{\text{F}}_{n})} \\ L^{\text{W}}_{\text{T}} = -\dfrac{1}{K}\displaystyle\sum_{k=1}^{K} \log \dfrac{\exp((P^{\text{W}}_{t+k})^{\text{T}} z^{\text{W}}_{t+k})}{\sum_{n\in[t,k]} \exp((P^{\text{W}}_{t+k})^{\text{T}} z^{\text{W}}_{n})} \end{cases} \tag{3}$$

At the same time, c_t^{W} and c_t^{F} from the same sample are regarded as a positive sample pair c_t^{+}, and other context features from different samples are regarded as a negative sample pair c_t^{-}. The context loss function L_{C} can be constructed, and the overall loss is calculated as shown in Eq. (4) where τ is the temperature parameter, λ_1 and λ_2 are the weight parameters, and $\text{Sim}_{\cos}(\cdot)$ means cosine similarity function.

$$\begin{cases} L_{\text{C}} = -\displaystyle\sum_{i=1}^{N} \log \dfrac{\exp(\text{Sim}_{\cos}(c_t^{+})/\tau)}{\displaystyle\sum_{m=1}^{2N} \exp(\text{Sim}_{\cos}(c_t^{-})/\tau)} \\ L = \lambda_1 (L^{\text{F}}_{\text{T}} + L^{\text{W}}_{\text{T}}) + \lambda_2 L_{\text{C}} \end{cases} \tag{4}$$

Combining the above modules, the method flow of our proposed model is shown in algorithm 1.

3 Experimental Results

3.1 Experimental Environment

1) Dataset description

Human Activity Recognition (HAR): HAR[21] dataset contains embedded inertial sensors data from about 30 subjects doing 6 classes of Activities of Daily Living (ADL): walking, walking upstairs, walking downstairs, sitting, standing, and laying. Since there are sensors on each person recording data, the channel of time series data is 9 and the length of time series data is 128.

SleepEDF: A dataset about Electroencephalogram (EEG) signals in PhysioBank[22]. This dataset contains data from two experiments: One is the effect of age on sleep, and the other is the effect of temazepam on sleep.

Algorithm 1 Our model

Input:	Time series data X, model hyperparameter: output length k, weight λ_1 and λ_2
Output:	The learned multi-layer LSTM encoder
1:	Randomly initialize the multi-layer LSTM encoder as LSTMencoder()
2:	Randomly initialize the contrastive Transformer module as TransformerEncoder() and TransformerDecoder()
3:	Get augmentation data by X^{F} = FFT_augmentation(X) and $\boldsymbol{X}^{\mathrm{W}}$ = weak_augmentation(X)
4:	**For** epoch = 1 to Maximum **do**
5:	Representation voctor $\boldsymbol{Z}^{\mathrm{F}}$ = LSTMencoder(X^{F}) and $\boldsymbol{Z}^{\mathrm{W}}$ = LSTMencoder(X^{W})
6:	Randomly initialize the variable t
7:	Divide vector $\boldsymbol{Z}$ with variable t : $\boldsymbol{Z}^{\mathrm{W}}_{\mathrm{input}} = \boldsymbol{Z}^{\mathrm{W}}[0{:}t{+}1]$, $\boldsymbol{Z}^{\mathrm{F}}_{\mathrm{input}} = \boldsymbol{Z}^{\mathrm{F}}[0{:}t{+}1]$ $\boldsymbol{Z}^{\mathrm{W}}_{\mathrm{output}} = \boldsymbol{Z}^{\mathrm{W}}[t{+}1{:}t{+}k]$, $\boldsymbol{Z}^{\mathrm{F}}_{\mathrm{output}} = \boldsymbol{Z}^{\mathrm{F}}[t{+}1{:}t{+}k]$
8:	Get hidden feature $\boldsymbol{Z}_{\mathrm{input}}$ and contextual feature C_t C^{W}_t, $\boldsymbol{Z}^{\mathrm{W}}_{\mathrm{hidden}}$= TransformerEncoder($\boldsymbol{Z}^{\mathrm{W}}_{\mathrm{input}}$) and C^{F}_t, $\boldsymbol{Z}^{\mathrm{F}}_{\mathrm{hidden}}$= TransformerEncoder($\boldsymbol{Z}^{\mathrm{F}}_{\mathrm{input}}$)
9:	The decoder result $\boldsymbol{P}$ gets by contrastive $\boldsymbol{Z}$ $\boldsymbol{P}^{\mathrm{W}}$ = TransformerDecoder($\boldsymbol{Z}^{\mathrm{W}}_{\mathrm{hidden}}$, $\boldsymbol{Z}^{\mathrm{F}}_{\mathrm{output}}$) and $\boldsymbol{P}^{\mathrm{F}}$ = TransformerDecoder($\boldsymbol{Z}^{\mathrm{F}}_{\mathrm{hidden}}$, $\boldsymbol{Z}^{\mathrm{W}}_{\mathrm{output}}$)
10:	Get temporal loss $L^{\mathrm{W}}_{\mathrm{T}}$ by $\boldsymbol{P}^{\mathrm{W}}$, $\boldsymbol{Z}^{\mathrm{W}}_{\mathrm{output}}$ and $L^{\mathrm{F}}_{\mathrm{T}}$ by $\boldsymbol{P}^{\mathrm{F}}$, $\boldsymbol{Z}^{\mathrm{F}}_{\mathrm{output}}$
11:	Get contextual loss L_{C} by positive sample C^{W}_t and C^{F}_t and negative sample in $\boldsymbol{X}$
12:	Total loss $L = \lambda_1 * (L^{\mathrm{W}}_{\mathrm{T}} + L^{\mathrm{F}}_{\mathrm{T}}) + \lambda_2 * L_{\mathrm{C}}$
13:	Update model by L
14:	Get trained multi-layer LSTM encoder

Five classes are used to represent the subject's sleep state: wake, non-rapid eye movement which has three substates, and rapid eye movement. The length of the time series is 3 000.

Epilepsy: A dataset of surface EEG recordings from healthy volunteers with eyes closed and eyes opened[23]. It is divided into two categories, with epilepsy and without epilepsy. The length of the dataset is 5 120.

2) Running environment

The running environment is the same as TS-TCC[11]. The data is divided into training set, validation set, and test set according to the ratio of 3∶1∶1. The epoch of the self-supervised training is 100. We used Adam optimizer with a learning rate of $3E^{-4}$, weight decay of $3E^{-4}$, $\beta 1$ = 0.9, and $\beta 2$ = 0.99. Batch size is 128, the maximum segment M in HAR is 8, sleepEDF is 12, and Epilepsy is 5. The temperature parameter is 0.2, and the number of contrastive Transformer layers is 4. The weight parameter λ_1 is 1 and λ_2 is 0.7. The parameters α of the beta distribution in the FFT augmentation module in HAR is 0.5, sleepEDF is 0.5, and Epilepsy is 1. We have run our model on Pytorch 1.10.1 with CUDA 11.3 and it is trained on NVIDIA GeForce RTX 3060 GPU.

3.2 Overall Performance and Discussion

We test our model on HAR, sleepEDF, and Epilepsy datasets, and compare the accuracy rate between SSL-ECG[12], CPC[14], SimCLR[24], and TS-TCC[11]. After getting the representation vector from self-supervised learning, we will use one FC layer to classify time series data and calculate the accuracy. The model will be trained 5 times with 5 seeds and show the mean and standard deviation. The records of other models all come from Eldele[11]. The accuracy of the models is shown in Table 1.

The average accuracy rate can be used to represent the model's ability to represent and classify the data. As can be seen from Table 1, on the HAR and sleepEDF da-

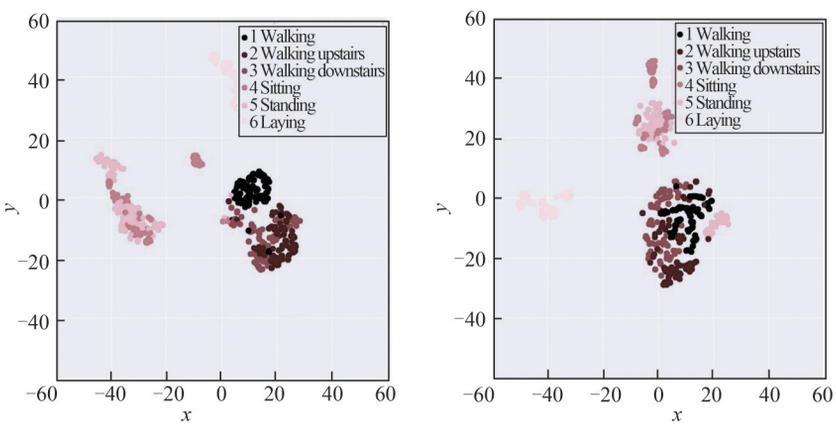
Table 1 Different model accuracy on three datasets %

Baseline	HAR	SleepEDF	Epilepsy
SSL-ECG	65.34±1.63	74.58±0.60	93.72±0.45
CPC	83.85±1.51	82.82±1.68	96.61±0.43
SimCLR	80.97±2.46	78.91±3.11	96.05±0.34
TS-TCC	90.37±0.34	83.00±0.71	**97.23±0.10**
Our model	**91.13±0.31**	**83.69±0.17**	96.98±0.34

tasets, the average accuracy of our model reaches 91.13% and 83.69%, which is higher than the average accuracy of all other models. However, on the Epilepsy dataset, our model accuracy is 96.98%, which is lower than 97.23% of the TS-TCC model, but higher than other models.

The standard deviation of the accuracy rate reflects the stability of the model under different random numbers. The smaller the standard deviation of the accuracy rate, the more stable the model, and the less the random number interferes with the model. Our model has the smallest standard deviation on both the HAR and sleepEDF datasets, 0.31 and 0.17. Especially on the sleepEDF dataset, the standard deviation of the accuracy of our model is much lower than that of other models. The above experiments show that our model has high accuracy, good stability and robustness in most cases.

In order to more clearly reflect the effect of time series data representation, the t-distributed stochastic neighbor embedding(t-SNE) [25] method maps the representation data from d dimensions to 2 dimensions to make a scatter plot as shown in Fig. 5.



(a) Our model feature (b) TS-TCC model feature

Fig. 5 Representation vector visualization on HAR using t-SNE

In Fig. 5, the closer the points of the same class, the higher the degree of representation of the model for the class, the more obvious the point features of the class for distinguishing from other categories, indicating that the representation vector contains more hidden features. It can be seen from Fig. 5 that the fifth class is the easiest to distinguish, and the third and fourth are the most difficult. Our model has certain advantages over TS-TCC in classifying class 1 because our class 1 is relatively farther away from classes 2 and 3 showing that our representation vector works better. Table 2 shows the precision and recall of our model and TS-TCC model for each class on the HAR dataset with self-supervised learning, verifying the effect of the t-SNE method.

Table 2 Precision and recall rate of our model and TS-TCC model on HAR with self-supervised learning

%

Baseline (self-supervised)	Quota	Class					
		1	2	3	4	5	6
Our model	Precision	94.05	96.30	98.05	89.48	78.65	98.90
	Recall	98.79	94.06	95.71	71.08	92.11	100
TS-TCC	Precision	98.46	91.49	88.19	79.79	80.33	99.80
	Recall	90.12	93.63	99.52	78.00	81.39	96.27

It can also be seen from Table 2 that the sixth class has the highest precision and recall rate, While the fourth and fifth categories are very low, which is also consistent with the results of t-SNE. From the recall rate, the first class of HAR can be classified well with other classes. In the t-SNE scatter plot, our first and second classes are close to spherical and denser than the TS-TCC model, so the recall rate of these two classes in Table 2 is also higher.

3.3 Test on Encoder Performance

Higher accuracy rate on the data set means a better

representation ability of the feature vector for the original data, and better effect of the encoder. In this experiment, to verify the effectiveness of the encoder module, we compare the effects of self-supervised and supervised training modes of encoder module of our model and TS-TCC model on the accuracy of the results of the three datasets: HAR, sleepEDF, and Epilepsy. Self-supervised training first learns the representation vectors through the encoder, and then classifies the feature vectors through one FC layer. When the FC layer is trained, the parameters in the encoder are frozen, that is to say, the representation vector remains unchanged, and only the FC layer is trained. However, in supervised training, the encoder is trained with the FC layer. The result is shown in Table 3.

Table 3 Encoder performance on TS-TCC and our model %

Baseline	HAR	Sleep-EDF	Epilepsy
TS-TCC-supervised	90.14±2.49	83.41±1.44	96.66±0.24
TS-TCC-self-supervised	90.37±0.34	83.00±0.71	97.23±0.10
Our model-supervised	**93.53±0.98**	**83.99±0.31**	**98.30±0.24**
Our model-self-supervised	91.13±0.31	83.69±0.17	96.98±0.34

As can be seen from Table 3, in supervised learning, the accuracy of our model is higher than that of the TS-TCC model on all three datasets, which indicates that the encoder module of our model has stronger representation ability than the encoder of TS-TCC model. When applied to the unsupervised domain, our model still outperformed the TS-TCC model in terms of accuracy on HAR and sleepEDF datasets. From the above comparison, it can be seen that the encoder in this paper provides better representation in both unsupervised and supervised domains. Table 4 specifically reflects the classification performance of the encoder for each class of the HAR dataset with supervised learning.

As can be seen from Table 4, the multi-layer LSTM encoder module can learn more hidden information about the first type of data. In Table 4, the accuracy of our model on the first class is much higher than that of TS-TCC indicating the higher ability of the proposed multi-layer LSTM encoder module in learning more hidden information about the first type of data, which is consistent with the result of Table 2. Moreover, we can find that supervised learning always represents higher

Table 4 Precision and recall rate of our model and TS-TCC model on HAR with supervised learning %

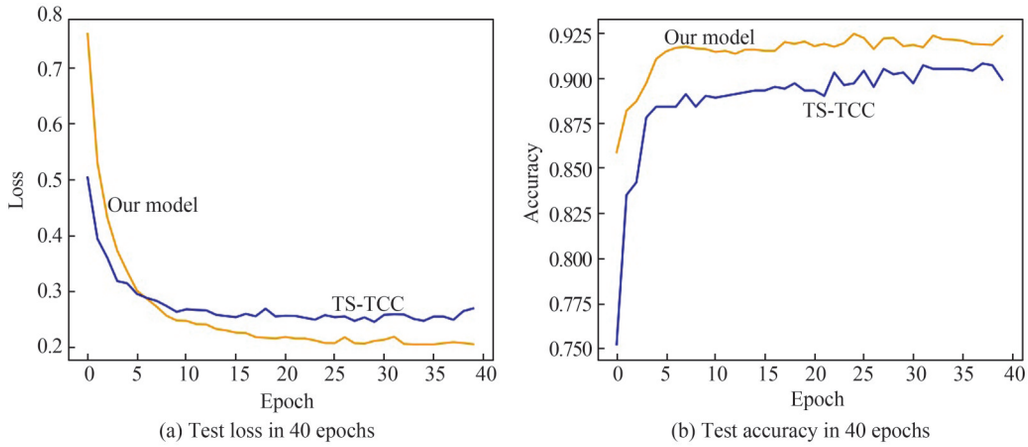
Baseline (supervised)	Quota	Class					
		1	2	3	4	5	6
Our model	Precision	100.00	99.57	97.45	82.00	87.90	98.35
	Recall	99.60	97.45	100.00	85.34	83.27	100.00
TS-TCC	Precision	98.97	88.78	99.27	86.64	82.38	99.42
	Recall	96.77	99.15	96.90	76.58	89.66	94.97

classification accuracy in each class than unsupervised learning in our model and TS-TCC model. This may be due to the fact that the encoder used in the unsupervised field is easy to ignore some details that can only be noticed by supervision.

In order to intuitively reflect the changing trend of the accuracy rate when the pre-trained representation vectors of the two models classify the time series data, the test loss and accuracy rate on the test set are shown in Fig. 6. As can be seen from Fig. 6(a), the TS-TCC model converged faster than our model during the training process. The converge starts in the 8th epoch for TS-TCC model, but 20th epoch for our model. Furthermore, the training loss of our model is smaller than that of TS-TCC model from the fifth epoch. This may be because the representation vector we learn contains more hidden feature, and it is more difficult for the FC layer to converge. From Fig. 6(b), The accuracy curves of the two models become flat from the fifth epoch, indicating that the representation features obtained from the two models learn the differences between the classes of the dataset. The accuracy curves of the two models start to flatten out at the fifth epoch, which indicates that the FC layer can easily classify most of the data correctly, and also indicates that the encoder module learns many classification features of the data. Meanwhile, our accuracy curves are always higher than the TS-TCC model, which indicates that our model can learn better than the TS-TCC model in the same short time as TS-TCC does.

3.4 FFT Augmentation and Contrastive Transformer Sensitivity Testing

This experiment tests the performance of the model under different hyperparameters on the HAR dataset, which includes the parameters α of the beta distribution in the FFT augmentation module and the weight parameters λ_1 and λ_2. As can be seen in Eq. (4), λ_1 and λ_2 respectively represent the influence of temporal loss and con-



(a) Test loss in 40 epochs (b) Test accuracy in 40 epochs

Fig. 6 Test loss and test accuracy of our model and TS-TCC

textual loss in loss function of the contrastive Transformer module. The result is shown in Fig. 7.

As can be seen from Fig. 7(a), the accuracy fluctuates above the dotted line (90.31%, TS-TCC) with the change of parameters α, meaning the parameters α of the beta distribution in the FFT enhancement have less influence on the overall experimental results. From Fig. 7(b), when the value of λ_1 is small, the accuracy curve tends to level off and remains above 90%, which indicates that the model can provide high accuracy even when the temporal loss has little effect on the loss function. Temporal loss may actually play an auxiliary role in the model, which is used to improve the model's representation of hidden feature that is difficult to learn. Figure 7(c) shows that when the value of λ_2 is small, the accuracy of the model increases with the value of λ_2. After the value of λ_2 exceeds 1, the change in accuracy tends to level off. This result indicates that the effect of the model is strongly related to the presence or absence of contextual loss. When the proportion of contextual loss in the loss function is too small, the performance of the model is poor, and simply increasing the contextual loss will not make the model effect continue to rise, but remains constant at a certain value. The above results show that our model is insensitive to the parameters within a certain range.

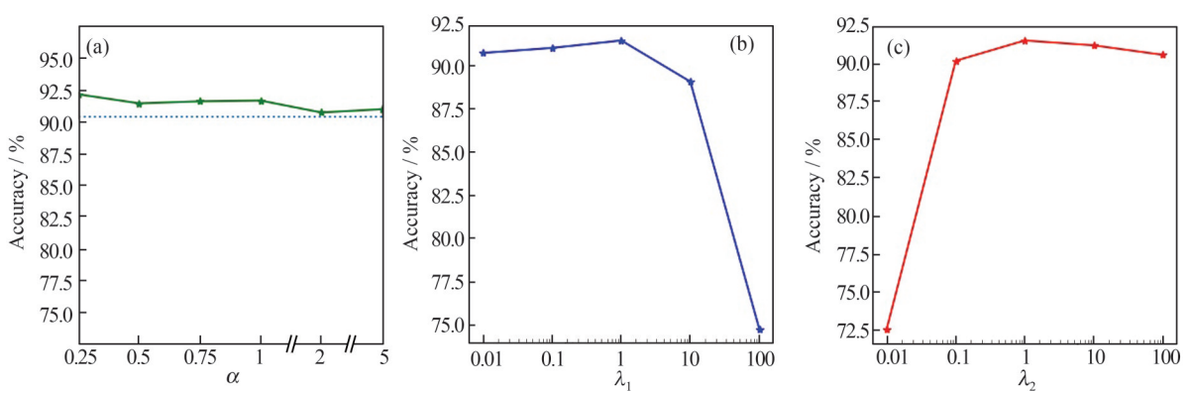


Fig. 7 Hyperparameter experiment result of parameters α (a), λ_1(b), λ_2(c)

4 Conclusion

This paper proposes an improved model based on TS-TCC. This model uses an FFT-based data augmentation method. The positive sample pairs are obtained by different augmentations to the original data, and the encoder is used to convert the data into a representation vector. In order to obtain more hidden features, our model uses a contrastive Transformer architecture and context loss. In this paper, our proposed model is tested with other time series contrastive learning models on three datasets, HAR, sleepEDF, Epilepsy, based on self-

supervised learning for classification. The experimental result show that except for the Epilepsy dataset where it is slightly lower than the TS-TCC model, our model is more accurate and stable than the other models on the three datasets. In the experiments with supervised and unsupervised learning, our designed multi-layer LSTM encoder module can indeed learn the hidden feature of some indistinguishable classes. Moreover, the sensitivity test shows our designed module is insensitive to hyper-parameters.

References

[1] Aghabozorgi S, Seyed Shirkhorshidi A, Ying Wah T. Time-series clustering—A decade review [J]. *Information Systems*, 2015, **53**: 16-38.

[2] Rabiner L, Juang B. An introduction to hidden Markov models [J]. *IEEE ASSP Magazine*, 1986, **3**(1): 4-16.

[3] Luenberger D G. *Theory*, *Models*, *and Applications* [M]. New York: Wiley, 1979.

[4] Wang X Y, Mueen A, Ding H, *et al*. Experimental comparison of representation methods and distance measures for time series data [J]. *Data Mining and Knowledge Discovery*, 2013, **26**(2): 275-309.

[5] Längkvist M, Karlsson L, Loutfi A. A review of unsupervised feature learning and deep learning for time-series modeling [J]. *Pattern Recognition Letters*, 2014, **42**: 11-24.

[6] Wang H S, Zhang Q, Wu J, *et al*. Time series feature learning with labeled and unlabeled data [J]. *Pattern Recognition*, 2019, **89**: 55-66.

[7] Jing L L, Tian Y L. Self-supervised visual feature learning with deep neural networks: A survey [J]. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 2021, **43**(11): 4037-4058.

[8] Doersch C, Gupta A, Efros A A. Unsupervised visual representation learning by context prediction [C]// *IEEE International Conference on Computer Vision*. New York: IEEE, 2015: 1422-1430.

[9] Gidaris S, Singh P, Komodakis N. Unsupervised representation learning by predicting image rotations [EB/OL]. [2022-09-11]. *https*://*arxiv.org/abs/*1803.07728.

[10] Jaiswal A, Babu A R, Zadeh M Z, *et al*. A survey on contrastive self-supervised learning [J]. *Technologies*, 2020, **9**(1): 2.

[11] Eldele E, Ragab M, Chen Z H, *et al*. Time-series representation learning via temporal and contextual contrasting [EB/OL]. [2022-09-11]. *https*://*arxiv.org/abs/*2106.14112.

[12] Sarkar P, Etemad A. Self-supervised ECG representation learning for emotion recognition [J]. *IEEE Transactions on Affective Computing*, 2022, **13**(3): 1541-1554.

[13] Löwe S, O'Connor P, Veeling B S. Putting an end to end-to-end: Gradient-isolated learning of representations [EB/OL]. [2022-09-11]. *https*://*arxiv.org/abs/*1905.11786.

[14] Oord A V D, Li Y Z, Vinyals O. Representation learning with contrastive predictive coding [EB/OL]. [2022-09-11]. *https*://*arxiv.org/abs/*1807.03748.

[15] Franceschi J Y, Dieuleveut A, Jaggi M. Unsupervised scalable representation learning for multivariate time series [EB/OL]. [2022-08-09]. *https*://*arxiv.org/abs/*1901.10738.

[16] Tonekaboni S, Eytan D, Goldenberg A. Unsupervised representation learning for time series with temporal neighborhood coding [EB/OL]. [2022-09-20]. *https*://*arxiv. org/abs/* 2106.00750.

[17] Iwana B K, Uchida S. An empirical survey of data augmentation for time series classification with neural networks [J]. *PLoS One*, 2021, **16**(7): e0254841.

[18] Zhou H Y, Zhang S H, Peng J Q, *et al*. Informer: Beyond efficient transformer for long sequence time-series forecasting [J]. *Proceedings of the AAAI Conference on Artificial Intelligence*, 2021, **35**(12): 11106-11115.

[19] Bazi Y, Bashmal L, Rahhal M M A, *et al*. Vision transformers for remote sensing image classification [J]. *Remote Sensing*, 2021, **13**(3): 516.

[20] Devlin J, Chang M W, Lee K, *et al*. BERT: Pre-training of deep bidirectional transformers for language understanding [EB/OL]. [2022-10-09]. *https*://*arxiv.org/abs/*1810.04805.

[21] Anguita D, Ghio A, Oneto L, *et al*. A public domain dataset for human activity recognition using smartphones[J]. 21*st European Symposium on Artificial Neural Networks*, *Computational Intelligence and Machine Learning*, 2013: 437-442.

[22] Goldberger A L, Amaral L A N, Glass L, *et al*. PhysioBank, PhysioToolkit, and PhysioNet [J]. *Circulation*, 2000, **101** (23): e215-e220.

[23] Andrzejak R G, Lehnertz K, Mormann F, *et al*. Indications of nonlinear deterministic and finite-dimensional structures in time series of brain electrical activity: Dependence on recording region and brain state [J]. *Physical Review E*, 2001, **64**(6): 061907.

[24] Chen T, Kornblith S, Norouzi M, *et al*. A simple framework for contrastive learning of visual representations [C]// *Proceedings of the* 37*th International Conference on Machine Learning*. New York: ACM, 2020: 1597-1607.

[25] van der Maaten L, Hinton G. Visualizing data using t-SNE [J]. *Journal of Machine Learning Research*, 2008, **9**(11): 2579-260.

□