A T-CNN Time Series Classification Method

Junlu Wang

Liaoning University

Su Li

Liaoning University

Wanting Ji

Liaoning University

Tian Jiang

Liaoning University

Baoyan Song (✉ bysong@lnu.edu.cn)

Liaoning University

Research Article

Keywords: Gram matrix, T-CNN model, Toeplitz, Loss function, Triplet network

Posted Date: November 30th, 2021

DOI: <https://doi.org/10.21203/rs.3.rs-1107199/v1>

License: This work is licensed under a Creative Commons Attribution 4.0 International License.

[Read Full License](#)

A T-CNN Time Series Classification Method

Junlu Wang, Su Li, Wanting Ji, Tian Jiang, Baoyan Song*

School of Information, Liaoning University, Shenyang, 110036, China

wangjunlu@lnu.edu.cn liisuu@163.com wanting.ji@lnu.edu.cn 1447131424@qq.com
bysong@lnu.edu.cn

Abstract: Time series classification is a basic task in the field of streaming data event analysis and data mining. The existing time series classification methods have the problems of low classification accuracy and low efficiency. To solve these problems, this paper proposes a T-CNN time series classification method based on a Gram matrix. Specifically, we perform wavelet threshold denoising on time series to filter normal curve noise, and propose a lossless transformation method based on the Gram matrix, which converts the time series to the time domain image and retains all the information of events. Then, we propose an improved CNN time series classification method, which introduces the Toeplitz convolution kernel matrix into convolution layer calculation. Finally, we introduce a Triplet network to calculate the similarity between similar events and different classes of events, and optimize the squared loss function of CNN. The proposed T-CNN model can accelerate the convergence rate of gradient descent and improve classification accuracy. Experimental results show that, compared with the existing methods, our T-CNN time series classification method has great advantages in efficiency and accuracy.

Key words: Gram matrix; T-CNN model; Toeplitz; Loss function; Triplet network

1 Introduction

A time series is the result of a set of sequence data observed for a potential process at a given sampling rate in an equal period [1]. In recent years, with the development of computer technology, time series has become more and more widely applied, such as disaster monitoring, safety analysis, climate change, stock funds and other fields. How to classify time series [2-4] has been a difficult point in the field of data mining. For example, mine disaster monitoring and early warning systems use microseismic sensors deployed around mines to storage real-time data. These mine disaster events usually last from a few seconds to more than 10 seconds. Classifying these events helps to summarize the feature of various types of disasters [5], which is of great significance for data analysis and disaster prevention.

The existing time series classification methods mostly use symbolic aggregation approximation (SAX) [6] and convolution neural network (CNN) [7][8], but ignore the time attribute and the classification accuracy is not high. Therefore, to solve the above problems, this paper proposes a T-CNN time series classification method based on the Gram matrix [9]. The proposed method preserves the time attribute, improves the square loss function [10] of the CNN model in the full connection layer to enhance classification accuracy and efficiency. The main contributions of this paper are as follows:

- Aiming at losing time attributes, a transformation method based on the Gram matrix is proposed, which converts time series into time-domain images without loss, and uses the wavelet threshold denoising method to filter out normal background noise [11][12].
- In order to increase the network convergence rate, a Toeplitz convolution kernel matrix [13] is introduced into the CNN convolution layer, and the product of the Toeplitz matrix is used to replace the traditional convolution operation.

- A T-CNN model is proposed, which introduces a triplet network [14] in the fully connected layer to calculate the difference function between the same class and different classes, and optimizes the square loss function of the CNN model, to improve the classification accuracy.

2 Related work

At present, researchers have studied the time series classification and obtained some research achievements.

Reference [15] proposed SAX, which divided the series into segments and converted each segment into a corresponding letter for classification. This method used the idea of aggregation to effectively reduce the dimension and increase the classification efficiency. However, this method lost a large number of data points in the time series, and the classification accuracy of this method was not high. Reference [16] proposed a trend turning point (TTP) method, which extracted the trend feature of the time series itself and required statistical analysis of the extreme points and inflection points of each time series. This method had a good effect on the data sets with specific rules, but not good for the data sets with large trend fluctuations.

Reference [17] proposed a time domain distance (TDD) method, which reflected the similarity by calculating the Euclidean distance between different sequences. The closer the distance, the higher the similarity. The classification speed of this method was fast, but the classification accuracy was low. Reference [18] proposed an interval classification method, which divided time series into equal-length intervals, calculated the mean value and standard deviation of each interval, and used support vector machines for classification. This method regarded time series as a set of discrete data points and lost the time attribute.

Reference [19] proposed a Shapelet method to find the Shapelet from the time series of known class labels to reflect the corresponding classification features. This method has high classification accuracy, but the time complexity of Shapelet extraction is high, the classification efficiency is low. Reference [20] proposed a classification method based on convolution neural networks, which used CNNs to effectively classify time series. However, the training process of this method was complicated, and did not consider the time attribute.

To solve the above problems, this paper proposes a T-CNN time series classification method based on the Gram matrix, which converts time series into time-domain images without loss and retains time attributes. The proposed T-CNN model based on convolution neural networks inputs the time-domain images into the model, improving the classification accuracy.

3 Gram matrix conversion of time series

Time series events are not one or a few abnormal perception data points, but a series of discrete data points that meet the threshold range in the time domain, and the same class events have similar features.

Definition 1 Time series events: A collection of continuous abnormal data initiated by the first outlier in the time series that exceeds the threshold range and lasts for a period of time. It can be represented as:

$$E = \{(e_1, e_2, e_3, \dots, e_n) \mid \forall i \in (1, 2, 3, \dots, n), |e_i| > \varphi\}, \quad (1)$$

where e_i represents an outlier in an event, and φ denotes the given threshold range.

According to Definition 1, each time series event has time attributes. Aiming at the lack of time attribute in the existing time series classification methods, we introduce the Gram matrix

conversion method to realize the lossless conversion of time series.

3.1 Normal noise filtering

After Gram matrix conversion, the time-domain image histogram presents normal distribution. The existence of normal noise directly affects the conversion of images in the time domain. Therefore, before the Gram matrix is used to convert the time series, the data is preprocessed first, and then the wavelet threshold denoising method is used to filter out the normal background noise carried by the data.

The steps of the wavelet threshold denoising method are as follows:

(1) Wavelet decomposition of signals. A wavelet is selected to determine the S -level wavelet decomposition, and then the S -level wavelet decomposition calculation is carried out for the signals.

(2) Threshold quantization of high-frequency coefficients in wavelet decomposition. From layer 1 to layer S , a threshold is selected for the high-frequency coefficients of each layer for threshold quantization. The threshold function used in this paper is calculated by:

$$\lambda = \sigma \sqrt{2 \ln N}, \quad (2)$$

where $\sigma = M/0.6745$ denotes the median value of the absolute value of the first-level wavelet decomposition coefficient, 0.6745 is the adjustment coefficient of the Gaussian noise standard deviation, and N denotes the length of the signal.

(3) Wavelet reconstruction of signals. According to the low-frequency coefficients of the S layer and the quantized high-frequency coefficients of the first layer to the S -th layer of the wavelet decomposition, the signal is reconstructed by wavelet.

Similar time series events have the same features, but the locations and intensities of the events are different, and the perceived event data are not at the same scale. Therefore, the normalization equation is used to keep the weight of the data feature dimension on the objective function consistent. The data normalization equation is as follows:

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}}, \quad (3)$$

where x denotes the data to be normalized, x' denotes the normalized result, $x_{\min}$ represents the minimum value in the time series data, and $x_{\max}$ represents the maximum value in the time series data. Through data normalization conversion, all data can be normalized to [0,1]. For example, Fig. 1 shows a normalized time series $T' = \{t_1', t_2', t_3', \dots, t_n'\}$ calculated by Equation 3.

Fig.1 Time series.

3.2 Time domain image conversion based on Gram matrix

The time attribute is an important attribute to determine the classification. Due to the different time attributes, different regular distributions are presented in chronological order. Therefore, this paper proposes a time-domain image conversion based on the Gram matrix. This method can retain the time attributes and convert a time series into an $N \times N$ two-dimensional matrix without loss.

The Gram matrix G is a matrix composed of the inner product of each pair of vectors in the form of Equation 3:

$$G = \begin{pmatrix} \langle a_1, a_1 \rangle & \langle a_1, a_2 \rangle & \dots & \langle a_1, a_n \rangle \\ \langle a_2, a_1 \rangle & \langle a_2, a_2 \rangle & \dots & \langle a_2, a_n \rangle \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \langle a_n, a_1 \rangle & \langle a_n, a_2 \rangle & \dots & \langle a_n, a_n \rangle \end{pmatrix}, \quad (4)$$

where $\langle a_i, a_j \rangle$ denotes the inner product of two vectors.

The matrix G is a positive definite matrix. The proof of G is as follows:

Let

$$Z = (x_1, x_2, \dots, x_n)^T \neq 0,$$

then

$$Z'GZ = (x_1, x_2, \dots, x_n) \begin{pmatrix} \langle a_1, a_1 \rangle & \langle a_1, a_2 \rangle & \dots & \langle a_1, a_n \rangle \\ \langle a_2, a_1 \rangle & \langle a_2, a_2 \rangle & \dots & \langle a_2, a_n \rangle \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \langle a_n, a_1 \rangle & \langle a_n, a_2 \rangle & \dots & \langle a_n, a_n \rangle \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{pmatrix}.$$

Thus,

$$Z'GZ = \left(\sum_{i=1}^n a_i x_i, \sum_{i=1}^n a_i x_i \right) \geq 0,$$

where $Z'GZ$ represents a positive definite quadratic form, and G denotes a positive definite matrix. A positive definite matrix can retain the feature of the matrix by calculating the eigenvalues. After converted the time series by the Gram matrix, the time attributes can be retained. The result converted into the Gram matrix form is as follows:

$$G_t = \begin{pmatrix} \langle t_1, t_1 \rangle & \langle t_1, t_2 \rangle & \dots & \langle t_1, t_n \rangle \\ \langle t_2, t_1 \rangle & \langle t_2, t_2 \rangle & \dots & \langle t_2, t_n \rangle \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot & \cdot \\ \langle t_n, t_1 \rangle & \langle t_n, t_2 \rangle & \dots & \langle t_n, t_n \rangle \end{pmatrix}, \quad (5)$$

where G_t denotes the time series after Gram matrix conversion, $\langle t_i, t_j \rangle$ is the inner product pair of the time series, and n is the length of the time series. An inner product represents the correlation between two points, G_t a real symmetric matrix. From left to right of the first row and from top to bottom of the first column in the G_t , the correlation between the first point and the subsequent points of the time series increases with time. Similarly, the second row and the second column are the correlation between the second point and the subsequent points. Therefore, the G_t matrix from the upper left to the low right represents the order of correlation arrangement between two points with time increasing. The time attribute of the time series is retained in the Gram matrix.

Since the time series T is one-dimensional, and in the rectangular coordinate system, the inner product pair calculation requires two-dimensional information of the abscissa and the ordinate. Therefore, to better preserve the time attribute of the time series, we use the polar coordinate system to calculate the inner product pairs of the time series.

The time series T can be encoded into polar coordinates by Equations 6 and 7:

$$\theta_i = \arccos(t_i'), \quad (6)$$

$$r_i = \frac{i}{n}, \quad (7)$$

where t_i' denotes the normalized time series, i is the time stamp in the time series, n is the length of the time series, θ_i is the angle of a point in the time series in the polar coordinate system, and r_i is the radius of a point in the time series in the polar coordinate system. Thus, each point in the time series can be represented by Equations 6 and 7.

Fig. 2 shows the polar coordinate coding of T' in Fig. 1. The encoding process for 0.5 is:

According to Equation 6, $\arccos(0.5)=60^\circ$; $i=3$; $N=5$; according to Equation 7, the radius is 0.6. Thus, $(0.6, 60^\circ)$ is the coding result.

Fig. 2 Polar coordinates of time series. According to Equation 6, $\arccos(0.5)=60^\circ$; $i=3$; $N=5$; according to Equation 7, the radius is 0.6. Thus, $(0.6, 60^\circ)$ is the coding result.

As time goes on, the radius of the point becomes larger and larger, and gradually moves away from the center of the circle. Polar coding can completely preserve the time attribute by increasing the radius, and the numerical change can be represented by the angle change. Therefore, as shown in Equation 8, a new Gram matrix can be obtained by using the polar coordinate angle relationship between two points in the time series:

$$G_t = \begin{pmatrix} \cos(\theta_1 + \theta_1) & \cos(\theta_1 + \theta_2) & \dots & \cos(\theta_1 + \theta_n) \\ \cos(\theta_2 + \theta_1) & \cos(\theta_2 + \theta_2) & \dots & \cos(\theta_2 + \theta_n) \\ \vdots & \vdots & \ddots & \vdots \\ \cos(\theta_n + \theta_1) & \cos(\theta_n + \theta_2) & \dots & \cos(\theta_n + \theta_n) \end{pmatrix}, \quad (8)$$

where $\cos(\theta_i + \theta_j) = \cos \theta_i \cos \theta_j - \sin \theta_i \sin \theta_j = t_i \times t_j - \sqrt{1-t_i^2} \times \sqrt{1-t_j^2}$, and θ_i denotes the angle obtained by Equation 6. When $i = j$ is formed the diagonals $\cos(\theta_i + \theta_i) = 2t_i^2 - 1$ of the G_t matrix, and the diagonal of the G_t matrix is arranged in order.

As the position of the Gram matrix moves from the upper left corner to the lower right corner, the time series values are arranged into the matrix sequentially. In other words, we preserve the time attribute of the time series, and encode the time dimension into the geometric structure of the matrix. Each value of the matrix is equivalent to the pixel of the image, and each time series is converted into a time-domain image by the Gram matrix.

4 Time domain image T-CNN classification

As mentioned above, time series are converted to Gram time-domain images, and the Gram time-domain images are used as the input matrix of convolutional neural networks for classification. In order to solve the problems of complex computation and slow training speed of convolutional neural networks, we propose a method based on the Toeplitz matrix product to replace the convolution operation of the convolution layer, and introduce the idea of triplet network into the loss function to improve the efficiency and accuracy of classification.

4.1 Convolution based on Toeplitz matrix multiplication

The convolution operation based on the Toeplitz matrix product is shown in Fig. 3. In Fig. 3, the dark blue square represents the convolution kernel, and the light blue square represents the matrix to be convoluted. The convolution kernel is 2×2 , the unconvoluted matrix is 3×3 , the step size is 1. The traditional convolution is shown in the upper part of Fig. 3. The convolution kernel moves successively on the matrix to be convoluted according to the step size of 1, and it requires four traversals of the complete matrix to be convoluted. After each traversal, the convolution kernel and the matrix part with its repeated sum are multiplied and accumulated, and the obtained value is the local convolution result at the corresponding position. Since the traditional convolution needs to traverse the whole image, the computational complexity is high.

Fig.3 Convolutional computation.

As shown in the lower part of Fig. 3, based on the Toeplitz matrix product convolution, each 3×3 process matrix obtained by the convolution kernel traversal matrix is expanded in line with row order to obtain a $4 \times 1 \times 9$ row matrix, forming a large matrix H . Then, the convolution matrix is expanded into a 9×1 column vector X in line with row arrangement order. The product of the large matrix H constructed by the convolution kernel and the column vector X to be constructed by the convolution matrix effectively replaces the convolution computation. Specifically, the convolution kernel matrix H is composed of 6 small matrices, which are respectively the matrix in the red box and the matrix in the yellow box, as well as the zero matrix in the two white parts.

In Fig. 3, the matrix in the red box conforms to the definition form of the Toeplitz matrix. Similarly, the matrix in the yellow box and the zero matrix are Toeplitz matrices. Therefore, the convolution kernel matrix constructed is a large Toeplitz matrix composed of several small Toeplitz matrices. The product of the Toeplitz matrix is used to replace the traditional convolution operation. The convolution kernel is directly constructed into the convolution kernel matrix without traversing the image in order of step size, and the product of the two matrices is calculated to reduce the computational complexity.

Definition 2 Toeplitz matrix: A matrix with the same elements on each diagonal line from the top left to the bottom right is a Toeplitz matrix, which has the properties $A_{i,j} = A_{i+1,j+1} = a_{i-j}$. Mathematically,

$$A = \begin{pmatrix} a_0 & a_{-1} & a_{-2} & \dots & \dots & a_{-(n-1)} \\ a_1 & a_0 & a_{-1} & \ddots & & \vdots \\ a_2 & a_1 & \ddots & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & \ddots & a_{-1} & a_{-2} \\ \vdots & & \ddots & a_1 & a_0 & a_{-1} \\ a_{n-1} & \dots & \dots & a_2 & a_1 & a_0 \end{pmatrix} \quad (9)$$

4.1.1 Toeplitz convolution kernel matrix Construction

In order to replace the convolution calculation with the Toeplitz matrix multiplication operation, the convolution kernel matrix H is constructed as the Toeplitz convolution kernel matrix H_t . Given any convolution kernel matrix as follow:

$$H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1D} \\ h_{21} & h_{22} & \cdots & h_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ h_{C1} & h_{C2} & \cdots & h_{CD} \end{pmatrix}$$

The corresponding construction steps of the Toeplitz convolution kernel matrix are as follows:

(1) A small Toeplitz matrix is generated from each row element of the convolution kernel matrix. Since the size of the convolution kernel matrix is $C \times D$, the convolution kernel matrix H is divided into C Toeplitz matrices: $H_0, H_1, H_2, H_3, \dots, H_{C-1}$, where H_0 is the zero interpolation of the element h_{11} in the first row and first column of H , the number of inserted zeros is the number of columns in the convolution kernel matrix H minus 1, and the interpolation result is taken as the first row of H_0 . Then h_{12} is interpolated as the second row according to the properties of the Toeplitz matrix until the $2 \times D - 1$ rows are formed and the H_0 construction is completed. By analogy, H_i is the matrix obtained by interpolating the $i - 1$ row elements of H . For example, the

convolution kernel matrix is $H = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, then H is divided into two matrices $H_0 = \begin{bmatrix} 1 & 0 \\ 2 & 1 \\ 0 & 2 \end{bmatrix}$ and

$$H_1 = \begin{bmatrix} 3 & 0 \\ 4 & 3 \\ 0 & 4 \end{bmatrix}.$$

(2) The small Toeplitz matrix obtained in Step (1) is formed into a large Toeplitz matrix:

$$H_t = \begin{pmatrix} H_0 & 0 & \cdots & 0 & 0 \\ H_1 & H_0 & \ddots & \vdots & \vdots \\ H_2 & H_1 & \ddots & 0 & 0 \\ \vdots & H_2 & \ddots & H_0 & 0 \\ H_{C-2} & \vdots & \ddots & H_1 & H_0 \\ H_{C-1} & H_{C-2} & \vdots & H_2 & H_1 \\ 0 & H_{C-1} & H_{C-2} & \vdots & H_2 \\ 0 & 0 & H_{C-1} & H_{C-2} & \vdots \\ \vdots & \vdots & \vdots & H_{C-1} & H_{C-2} \\ 0 & 0 & 0 & \cdots & H_{C-1} \end{pmatrix}. \quad (10)$$

$$H_t = \begin{bmatrix} H_0 & 0 \\ H_1 & H_0 \\ 0 & H_1 \end{bmatrix}$$

In the example in Step (1), is obtained by Equation 10, where 0 represents a zero matrix of 3×2 .

4.1.2 Toeplitz Matrix Convolution

After obtaining the Toeplitz convolution kernel matrix from Section 4.1.1, the traditional convolution can be replaced by the Toeplitz matrix multiplication using Equation 11.

$$X * H = H_t \times X_T, \quad (11)$$

where $X = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1B} \\ x_{21} & x_{22} & \cdots & x_{2B} \\ \vdots & \vdots & \vdots & \vdots \\ x_{A1} & x_{A2} & \cdots & x_{AB} \end{pmatrix}$ denotes the matrix to be convolved,

$H = \begin{pmatrix} h_{11} & h_{12} & \cdots & h_{1D} \\ h_{21} & h_{22} & \cdots & h_{2D} \\ \vdots & \vdots & \vdots & \vdots \\ h_{C1} & h_{C2} & \cdots & h_{CD} \end{pmatrix}$ denotes the convolution kernel, H_t is the Toeplitz convolution kernel

matrix in Section 4.1.1, and X_T is the column vector obtained by arranging all the elements of X in row order. Using the full convolution method, the matrix to be convolved is filled with zeros, and the result returns all the data after convolution. The row number of the convolution result matrix is $M = A+C-1$ and the column number of the convolution result matrix is $N = B+D-1$.

For example, when $X = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix}$, then $X_T = [5 \ 6 \ 7 \ 8]^T$, and the results that use

$$X * H = \begin{bmatrix} 5 & 6 \\ 7 & 8 \end{bmatrix} * \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix} = \begin{bmatrix} 5 & 16 & 12 \\ 22 & 60 & 40 \\ 21 & 52 & 32 \end{bmatrix}$$

convolution calculation is

$$H_t \times X_T = \begin{bmatrix} H_0 & 0 \\ H_1 & H_0 \\ 0 & H_1 \end{bmatrix} \times [5 \ 6 \ 7 \ 8]^T$$

convolution operation based on the Toeplitz matrix is

$= [5 \ 16 \ 12 \ 22 \ 60 \ 40 \ 21 \ 52 \ 32]^T$. Then the calculated column vector is rewritten into a 3×3 matrix according to $M = A+C-1 = 3$ and $N = B+D-1 = 3$, which is the same as the results of the convolution calculation.

We use the Toeplitz matrix multiplication to effectively replace the convolution operation. In terms of time complexity, the input time-domain image size is $A \times B$, and the convolution kernel size is $C \times D$. The convolution operation requires the convolution kernel to continuously traverse the time domain image and calculate $A \times B \times C \times D$ times multiplication.

When using the Toeplitz matrix multiplication, it only needs to calculate the matrix multiplication once. It is realized from Fig. 3 that there are a large number of zeros in each row of the matrix which does not need to be calculated. Thus, the actual calculation of each row is $C \times D$, the row number is the time of convolution kernel traverses, and approximately multiply $A \times B \times C \times D$ times. Therefore, during a calculation, the calculation amount of the two methods is roughly the same. However, when a new time-domain image is inputted into the traditional convolution each time, there are a large number of shift operations in the calculation, which greatly increases the calculation time.

Although it takes some time to construct the Toeplitz matrix, Toeplitz matrix multiplication only needs to construct the corresponding Toeplitz matrix once according to the given convolution kernel, and then can directly perform the matrix multiply calculation on all the input time-domain images to obtain the convolution result. In this way, for the datasets with a large number of sample

sets and test sets, the convolution operation time will be greatly reduced.

4.2 T-CNN Model classification

When a CNN model is used for classification, its fully connected layers perform convergence operations, and a given loss function is required. In this paper, the Triplet network is introduced into the loss function, and then the T-CNN model is proposed.

Let the sample set of m samples is $\{(x^{(1)}, y^{(1)}), (x^{(2)}, y^{(2)}), \dots, (x^{(m)}, y^{(m)})\}$, there are n classes in these samples, which $y^{(i)}$ represents the expected output of $x^{(i)}$, and the loss function of CNNs is shown as Equation 12:

$$R(\omega, b) = \frac{1}{m} \sum_{i=1}^m \left(\frac{1}{2} \|p_{\omega, b}(x^i) - y^{(i)}\|^2 \right), \quad (12)$$

where ω is the weight of each neuron, b is the bias, and $p_{\omega, b}(x^i)$ is the actual output of the sample. The CNN model continuously adjusts the parameter ω and b by training to minimize $R(\omega, b)$.

The CNN uses the gradient descent method to adjust the parameter $R(\omega, b)$, as shown in Equations 13 and 14:

$$\omega_{ij} = \omega_{ij} - a \frac{\partial}{\partial \omega_{ij}} R(\omega, b), \quad (13)$$

$$b_{ij} = b_{ij} - a \frac{\partial}{\partial b_{ij}} R(\omega, b), \quad (14)$$

where a is the learning rate and $R(\omega, b)$ is the CNN loss function.

In order to improve the classification accuracy, the Triplet network is introduced into the CNN loss function for constraint, and a T-CNN model based on the Triplet loss function is proposed. The idea of the T-CNN model is to input three-time domain images at a time, two of which belong to the same class and one belongs to other classes. The T-CNN model can obtain the feature of the time domain images by training and can obtain the feature difference function L_1 of two-time domain images from the same class and the feature difference function L_2 of two-time domain images from different classes. Then L_1 and L_2 are used to adjust the parameters of the T-CNN model. L_1 and L_2 are shown in Equations 15 and 16 respectively:

$$L_1 = \frac{1}{2} \|p_{\omega, b}^{(t_1)} - p_{\omega, b}^{(t_2)}\|^2, \quad (15)$$

$$L_2 = \frac{1}{2} \min \|n_{\omega,b}^{(l)} - p_{\omega,b}^{(l_i)}\|^2, (i=1,2), \quad (16)$$

where $p_{\omega,b}^{(l_i)}$ is the output value of the same class and $n_{\omega,b}^{(l)}$ is the output value of the different classes. The image feature difference functions are shown in the adjustment Equation (17).

$$L_T = \max(0, L_1 - L_2 + \alpha), \quad (17)$$

where α represents the minimum distance of the difference function between different classes and between classes (set to 0.1 in this paper). In each reverse iteration, L_T gradually approaches zero. As shown in Fig. 4, when the feature difference function L_1 of the same class of images is greater than the feature difference function L_2 of different classes of images minus the parameter α , L_T is greater than zero, and the model is adjusted in reverse to make L_1 smaller and L_2 larger.

Fig. 4 Adjustment chart of feature difference.

In Fig. 4, A and P belong to the same class, while N does not belong to the same class as A and P. Before the adjustment, the distance between A and P is greater than that between A and N, and the difference function L_T is greater than zero. Thus, the model parameters need to be adjusted in reverse. After the adjustment, the distance between A and N becomes larger, while the distance between A and P becomes smaller.

According to Equations 15 and 16, in each reverse iteration, it can be seen that L_1 will make the feature difference of the same class smaller, while L_2 will make the feature difference of different classes larger. On this basis, a Triplet loss function is proposed as shown in Equation 17:

$$L(\omega, b) = R(\omega, b) + \alpha L_1 - \beta L_2, \quad (17)$$

where $R(\omega, b)$ denotes the CNN loss function, α and β are coefficients greater than zero. L_1 is the feature difference function of the same class, L_2 is the feature difference function of different classes. The values of α and β need to be further determined through experiments. Therefore, the new residual error of each layer by the backpropagation algorithm is as follows:

$$\omega_{ij} = \omega_{ij} - a \frac{\partial}{\partial \omega_{ij}} L(\omega, b), \quad (18)$$

$$b_{ij} = b_{ij} - \alpha \frac{\partial}{\partial b_{ij}} L(\omega, b) \quad (19)$$

The T-CNN model based on the Triplet network adds the feature difference function between the same class and the feature difference function between different classes into cross-entropy loss function, which is conducive to allow the parameters to extract features with larger differences more quickly in the process of parameter weight adjustment. The partial derivative of $L(\omega, b)$ can make the backpropagation residual calculation to obtain new parameters ω and b . Each iteration is more inclined to the direction of gradient descent, which can make the model converge faster and improve the classification efficiency.

5 Experiment

The experimental data set adopts 6 data sets of UCR2018 and microquake data sets, which contains three classes of time series event waveforms of mine microseismic signals. The dataset includes time series data of different sizes, lengths and categories. The size of the training set accounts for 40% of the total data set, and the model training framework is Tensorflow. The hardware and software environment of the experiment are shown in Table 1.

Table 1 Software and hardware environment.

CPU	Intel Core (TM) i7-7500U
Memory	8GB
Hard disk	1TB
OS	Windows 8.1 (64bit)
Programming language	Python

The experiment data sets are shown in Table 2.

Table 2 The experiment data sets.

Data sets	Train	Test	Class	Length
BME	72	180	3	128
ItalyPowerDemand	438	1096	2	24
SyntheticControl	240	600	6	60
Cricket_X	312	780	12	300
SwedishLeaf	450	1125	15	128
WordSynonyms	362	905	25	270
microquake	4000	10000	3	unequal

In order to prevent local over convergence, the convolutional neural network structure is equipped with two convolutional layers. The convolutional layer 1 uses a convolution kernel of 5×5 size, and the maximum pooling window size is 3×3. The convolutional layer 2 uses a convolution kernel of 3×3 size, and the maximum pooling window size is 2×2. The convolutional layer 2 is followed by a standard fully connected layer. The activation function of this layer uses ReLU which makes the CNN model converge quickly. The loss function uses the Triplet loss function. The time-domain image output by the T-CNN model is $p = \{p_1, p_2, p_3\}$, where p_1, p_2, p_3 represent the probabilities of three classes in the data set respectively. $class = T(max(p))$ is used for judgment classes, where the function T is a threshold function, and the threshold is set to 0.8 after repeated experiments. When $max(p)$ is greater than the threshold set by the function T, the class of time-domain image is output. In order to make the experimental results more reliable, the experiment

uses a 10-fold cross validation, and the final result is an average of 10 times.

5.1 Comparison of model iteration times

The T-CNN model adjusts itself to the optimal state through continuous forward conduction, in which the main adjustable parameters are the learning rate and the iteration times. During this experiment, the learning rate is set to 0.005, and the iteration times is adjusted. Fig. 5 shows the change of time series classification accuracy under different iteration times. Among them, the horizontal axis is the iteration times, and the vertical axis is the classification accuracy.

Fig. 5 Comparison of model iteration number.

From Fig. 5, it is known that when the iteration time is less than 40 times, the classification accuracy increases faster, the classification accuracy increases slower from 40 to 70 times, and the classification accuracy is the highest at 70 times. It is concluded that increasing the iteration times within a certain range can improve the classification accuracy, but exceeding the range will cause the model over-fitting.

5.2 Comparison of classification accuracy

The accuracy is the proportion of the number of samples correctly classified to the total number of samples. Fig. 6 is the comparison of classification accuracy by seven different classification models. The compared methods are symbolic aggregation approximation (SAX), Shapelet, Dynamic Time Warping (DTW), Collective of Transformation-based Ensembles (COTE), Complexity Invariant distance (CID) and CNN classification method.

Fig. 6 Comparison of classification accuracy.

From Fig. 6, it is known that the T-CNN model converts the time series into the time domain images by using the Gram matrix, it can completely retain the time feature of the time series, and its classification accuracy is significantly better than other methods.

5.3 Comparison of classification precision

The precision is the proportion of samples predicted to be one class that is truly in that class, which is given by:

$$P_i = \frac{n_{ii}}{\sum_{j=1}^{n_c} n_{ji}}, \quad (20)$$

where n_{ij} is the number of samples predicted as the j th class in class i , and n_c is the number of sample classes. Then calculate the average precision of all classes to get the average precision of the classification. Fig. 7 is the comparison of classification precision.

Fig. 7 Comparison of classification precision.

From Fig. 7, due to the improved loss function of the T-CNN model, the classification precision is significantly better than other methods.

5.4 Comparison of classification recall

The recall is the proportion of samples detected in one class, which is given by:

$$R_i = \frac{n_{ii}}{\sum_{j=1}^{n_c} n_{ij}}, \quad (21)$$

where n_{ij} is the number of samples predicted as the j -th class in class i , and n_c is the number of sample classes. Then calculate the average recall of all classes to get the average recall of the classification. Fig. 8 is the comparison of classification recall.

Fig. 8 Comparison of classification recall.

From Fig. 8, the T-CNN model uses a Gram matrix to convert the time series without loss and improves the loss function, so the classification recall is significantly better than other methods.

6 Conclusion

This paper proposes a T-CNN time series classification method based on the Gram matrix. The method firstly denoises the time series by using wavelet threshold, and then converts the loss-free time series into time images by the Gram matrix, and finally inputs the time domain images to the proposed T-CNN model for classification. The proposed T-CNN model introduces the Toeplitz convolutional kernel matrix into the convolution layer, replaces the convolutional operation with the multiplication of two matrices, and introduces the feature difference function of the same class and different classes to improve the CNN loss function of the fully connected layer. Experimental results show that the proposed T-CNN time series classification method is suitable for processing time series classification, and the classification accuracy, precision and recall are better than the existing methods.

References:

1. U. Mori, A. Mendiburu, I.M. Miranda, J.A. Lozano. Early classification of time series using multi-objective optimization techniques[J]. Information Sciences, 2019, 492.
2. ZHAO C, WANG TJ, LIU SJ, et al. Rapid shapelet discovery algorithm for fusion selection extraction and subclass clustering [J]. Journal of Software, 2020, 31(03):763-777.
3. Houshang Darabi, Georgiana Ifrim, Diego Furtado Silva, et al. Guest editorial for special issue on time series classification[J]. IEEE/CAA Journal of Automatica Sinica, 2019, 6(06):1291-1292.
4. LI GL, YAN WH, WU ZD. Discovering shapelets with key points in time series classification[J]. Expert Systems with Applications, 2019, 132.
5. LUO H, SUN KX, WANG JL, et al. Multistage identification method for real-time abnormal events of streaming data.[J].International Journal of Distributed Sensor Networks, 2019, 15(12).
6. Hamdi Yahyaoui, Reem Al-Daihani. A novel trend based SAX reduction technique for time series[J]. Expert Systems With Applications, 2019, 130.
7. ZHOU FY, JIN LP, DONG J. A review of convolutional neural networks[J]. Chinese Journal of Computers, 2017, 40(06):1229-1251.
8. ZHAO BD, LU HZ, WU DY, et al. Convolutional neural networks for time series classification[J]. Journal of Systems Engineering and Electronics, 2017, 28(01):162-169.
9. WANG WG, YANG P, JIANG YL. Reduction method of bilateral H2 optimal model based on cross gram matrix [J]. Computer Engineering and Science, 2017(12):2203-2209.

10. ZHAO JP, GUO WW, ZHANG ZH, et al. A coupled convolutional neural network for small and densely clustered ship detection in SAR images[J]. Science China, 2019, 62(04):111-126.
11. ZHOU XC, WU T, SHI LF, et al. A wavelet transform image denoise method based on curvature variational regularization [J]. Acta Electronica Sinica, 2018, 46(03):621-628.
12. F á bio M. Bayer, Alice J. Kozakevicius, Renato J. Cintra. An iterative wavelet threshold for signal denoising[J]. Signal Processing, 2019, 162.
13. ZHENG YP, SHON SG, FU ZW. A gohberg-semencul type formula for the inverse of conjugate-Toeplitz matrix and applications[J]. Acta Mathematicae Applicate Sinica, 2018, 34(02):293-303.
14. HE GC, LIU XB. Unsupervised visual representation learning based on image triplet mining [J]. Chinese Journal of Computers, 2018, 41(12):2787-2803.
15. ZHONG QL, CAI ZX. An algorithm for symbolization of temporal data based on statistical features [J]. Chinese Journal of Computers, 2018(10):1857-1864.
16. ING J, SHI XD, SUN LY, et al. Data trend turning point extraction algorithm for time series [J]. Computer Engineering, 2018, 44(01):56-61.
17. Marion Morel, Catherine Achard, Richard Kulpa, S é verine Dubuisson. Time-series averaging using constrained dynamic time warping with tolerance[J]. Pattern Recognition, 2018, 74.
18. Marco A. Villegas, Diego J. Pedregal, Juan R. Trapero. A support vector machine for model selection in demand forecasting applications[J]. Computers & Industrial Engineering, 2018, 121.
19. Francisco J. Bald á n, Jos é M. Ben í tez. Distributed fastShapelet transform: A Big Data time series classification algorithm[J]. Information Sciences, 2019, 496.
20. CHEN W, SHI K. A deep learning framework for time series classification using relative position matrix and convolutional neural network[J]. Neurocomputing, 2019, 359.