Neurocomputing 359 (2019) 384–394

Contents lists available at ScienceDirect

Neurocomputing

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

# A deep learning framework for time series classification using Relative Position Matrix and Convolutional Neural Network

Wei Chen, Ke Shi*

*School of Computer Science and Technology, Huazhong University of Science and Technology, China*

ARTICLE INFO

*Article history:*
Received 26 September 2018
Revised 23 April 2019
Accepted 4 June 2019
Available online 13 June 2019

Communicated by Dr. Jie Wang

*Keywords:*
Time series classification
Deep learning
Convolutional Neural Network
Relative Position Matrix

ABSTRACT

Time series classification (TSC) which has attracted great attention in time series data mining task, has already applied to various fields. With the rapid development of Convolutional Neural Network (CNN), the CNN based methods on TSC have begun to emerge until recently. However, the performance of CNN based methods is slightly worse than state-of-the-art traditional methods. Therefore, we propose a novel deep learning framework using Relative Position Matrix and Convolutional Neural Network (RPMCNN) for the TSC task. We investigate a time series data representation method called Relative Position Matrix (RPM) to convert the raw time series data to 2D images which enable the use of techniques from image recognition. We also construct an improved CNN architecture to automatically learn a high-level abstract representation of low-level raw time series data. Therefore, the combination of RPM and CNN in a unified framework is expected to boost the accuracy and generalization ability of TSC. We conduct a comprehensive evaluation with various existing methods on a large number of standard datasets and demonstrate that our approach achieves remarkable results and outperforms the current best TSC approaches by a large margin.



## 1. Introduction

Temporal or sequential data is increasingly prevalent and important in analytics. We refer to such data sequences as time series if they are chronological sequences of observations and an important class of temporal data. Large volumes of time series naturally appear in almost every field such as biology, astronomy, meteorology, finance, medicine, engineering, and others [1]. The ubiquity of time series has generated an increasing level of interest in time series data mining tasks including query by content, clustering, classification, segmentation, prediction, anomaly detection, and motif discovery. Among all techniques applied to time series, time series classification (TSC) always has attracted great attention.

In recent years, a large number of new TSC algorithms for time series data have been developed [2]. Among these approaches, Bag of SFA Symbols (BOSS) [3] and Collective of Transformation Ensembles (COTE) [4] have shown better performance than others on the UCR datasets. However, it is becoming harder and harder to improve the accuracy by these traditional methods. Recently, as a result of the rapid development of more powerful computers, Deep Learning (DL) has emerged as a universal, practical and effective way to generate discriminative features of raw data. Therefore, it may offer clues to improve the accuracy of TSC methods. As one of the most successful DL model, Convolutional Neural Network (CNN) has been applied to solve sophisticated problems in many domains. The advantage of CNN is that it reduces the number of weights to make it easier to optimize the network and reduce the risk of overfitting to get better generalization ability based on the shared-weights architecture and translation invariance characteristics. Thus, CNN achieves remarkable successes in many tasks such as image and video recognition [5,6], recommender systems [7] and natural language processing [8].

Inspired by recent achievements of CNN based technologies in computer vision, we intend to treat the TSC task as an image recognition task. Specifically, the time series data is encoded as images first, and the salient features are automatically extracted by CNN model for classification subsequently. In this research, we propose a novel deep learning framework using Relative Position Matrix and Convolutional Neural Network (RPMCNN) for the TSC task. Specifically, a 2D representation method which is called Relative Position Matrix RPM (RPM) is proposed to convert the raw time series data to 2D images and an improved CNN model is proposed to classify these 2D images. The results show that our state-of-the-art approach achieves the best performance on 10 of 20 standard datasets from the UCR archive compared with 9 previous and current best classification methods.

* Corresponding author.
*E-mail addresses:* wchen_cs@huset.edu.cn (W. Chen), keshi@mail.hust.edu.cn (K. Shi).

https://doi.org/10.1016/j.neucom.2019.06.032
0925-2312/

The main contributions of this paper are summarized as the following three points. Firstly, a new simple and effective data representation method named as RPM is developed to generate the 2D images. Secondly, an improved CNN model using an appropriate structure is proposed to reduce the risk of overfitting. Meanwhile, it still holds a certain model complexity to obtain satisfied generalization ability. Thirdly, the proposed method achieves a significant improvement in accuracy compared with other traditional and state-of-the-art methods.

The rest of this paper is organized as follows. In Section 2 we discuss the background and related work. We introduce our framework in Section 3 and present the experimental results of the proposed method on the standard datasets in Section 4. Finally, the conclusions and future work are presented in Section 5.

## 2. Related work

The general literature on CNN is vast, however, it attracted huge attention immediately in 1995 [9]. Different from the fully connected artificial neural network (ANN), CNN is a special structure of ANN, each neuron of the feature map in each layer is sparsely connected to only a small set of neurons in the previous layer. This was inspired by the concept of simple and complex cells in the visual cortex of brain, and the visual cortex contains some cells that are only sensitive to the local receptive field. Moreover, CNN has led impressive progress in many fields including speech recognition, image recognition, and natural language processing in recent years, which motivates researchers to investigate CNN in TSC field.

Applications of CNN based methods on TSC have begun to emerge until recently, among these approaches, the most attractive ones are Multi-scale Convolutional Neural Network(MCNN) [10], Fully Convolutional Networks (FCN) [11], Gramian Angular Field (GAF)-Markov Transition Field (MTF) CNN model [12], and Recurrence Plots (RP) CNN model [13]. Furthermore, these approaches can be classified into two types according to the input data type of CNN, one includes MCNN and FCN which still uses 1D data after preprocessing of the raw time series data, and the other one includes GAF-MTF and RPCNN which converts the raw time series data to 2D images. Concretely, MCNN performs various transformations of the raw time series data by multi-scale branch and multi-frequency branch. The features of different branches are extracted by 1D convolutional layers and concatenated right after local convolution stage. Then all these features are fed into full convolution stage to obtain final results. MCNN addresses the limitation of previous research which only extracts features at a single time scale. However, the increase of input time series data will cause the increasing of useless information and larger memory consumption which may result in lower classification accuracy. FCN is proposed by Wang et al. [11] to build a standard baseline of applying Deep Neural Networks (DNN) in TSC field. In their research, three DNN models including Multilayer Perceptron (MLP) [14], FCN, and Residual Networks (ResNet) [15] are applied to TSC without any crafting in feature engineering and data preprocessing, which means the 1D raw time series data is directly fed to the DNN models. FCN achieves premium performance with simple protocol and small complexity for model building and deploying, which provides a strong baseline and a good starting point for future research. However, Wang and Oates [12] consider the problem of encoding time series as images to allow machines to visually recognize, classify and learn structures and patterns. They use GAF to transform the raw time series data into a polar coordinate system, and MTF to compute the transition probability in the manner of a first-order Markov chain along the time axis, which are treated as static and dynamic information of time series. A Tiled CNN [16] model is applied to classify the time series after GAF-MTF representation. However, the classification accuracy of their approach is not competitive compared to other state-of-the-art methods, which may be caused by the incomplete dynamic information that MTF offers. Moreover, Hatami et al. [13] propose a representation method based on RP [17] to convert the time series to 2D images with CNN model for TSC. In their research, time series are regarded as some distinct recurrent behaviors such as periodicities and irregular cyclicities which are the typical phenomenon of dynamic systems, and the main idea of using RP method is to reveal in which points some trajectories return to a previous state. A CNN model including two convolutional layers and two fully connected layers is applied to classify the images generated by RP. However, the result of their approach is unbalanced on the standard datasets due to the simple structure of the CNN model.

Recently, some advanced CNN models such as AlexNet [18], VGGNet [19], GoogLeNet [20], ResNet [15] and so on, have emerged since Lecun et al. [21] proposed the prototype of the CNN structure called LeNet-5. Among these famous models, VGGNet has shown better generalization ability on different tasks and datasets. Therefore, we propose a VGGNet based CNN model to solve the TSC task.

## 3. Methodology

### 3.1. RPMCNN Framework

The overall architecture of RPMCNN is depicted in Fig. 1, it consists of two sequential stages: representation stage and convolution stage.

1. In the representation stage, we apply a dimensionality reduction method first and then use RPM to transform the preprocessed time series into 2D images.
2. In the convolution stage, we construct an improved CNN architecture with several convolutional layers (each followed by max pooling), fully connected layers, and a softmax layer to generate the final output.

More details are described in Sections 3.2 and 3.3.

### 3.2. Representation stage

We denote a time series as $T = t_1, t_2, \ldots, t_n$, where $t_i$ is the value at time stamp $i$ and the time series length is $n$. Firstly, z-score normalization is applied to the raw time series data to obtain a standard normal distribution $Z = z_1, z_2, \ldots, z_n$ by :

$$z_i = \frac{t_i - \mu}{\sigma}, i = 1, 2, \ldots, n \tag{1}$$

where $\mu$ is the mean and $\sigma$ is the standard deviation of $T$. Then, we apply Piecewise Aggregation Approximation (PAA) [22] method to reduce the dimensionality of $Z$ to $m$. We choose an appropriate reduction factor $k$ to generate a new smooth time series as $X = x_1, x_2, \ldots, x_m$ by the following equation:

$$x_i = \begin{cases} \frac{1}{k}\sum_{j=k*(i-1)+1}^{k*i} z_j, i = 1, 2, \ldots, m, \left\lceil \frac{n}{k} \right\rceil - \left\lfloor \frac{n}{k} \right\rfloor = 0 \\ \begin{cases} \frac{1}{k}\sum_{j=k*(i-1)+1}^{k*i} z_j, i = 1, 2, \ldots, m-1 \\ \frac{1}{n-k*(m-1)}\sum_{j=k*(m-1)+1}^{n} z_j, i = m \end{cases}, \left\lceil \frac{n}{k} \right\rceil - \left\lfloor \frac{n}{k} \right\rfloor > 0 \end{cases}$$

$$m = \left\lceil \frac{n}{k} \right\rceil \tag{2}$$

where $m$ is the length of $X$. Simply stated, the normalized time series data is reduced from $n$ dimensions to $m$ dimensions while preserving the approximated trends of the original sequence by calculating the mean value of a piecewise constant. Subsequently, we construct a $m \times m$ matrix $M$ by RPM, which calculates the relative position between two time stamps to transform the preprocessed time series $X$ into 2D matrix. Each value at time stamp $i$ is treated

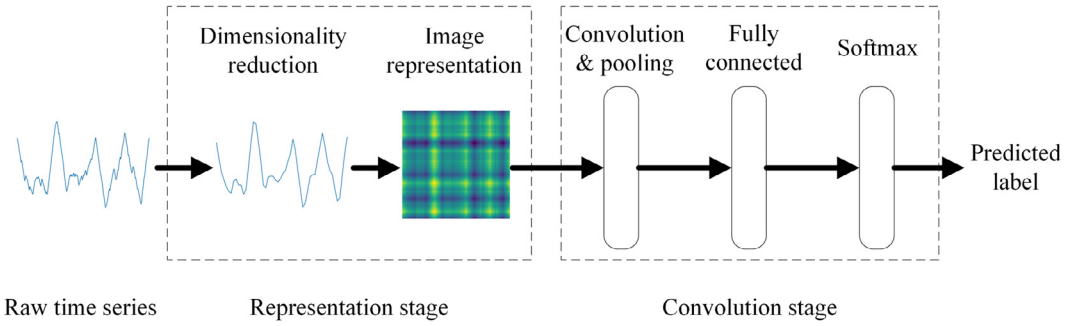
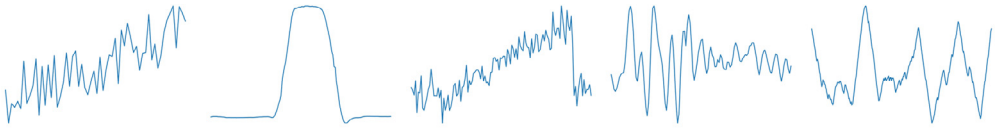
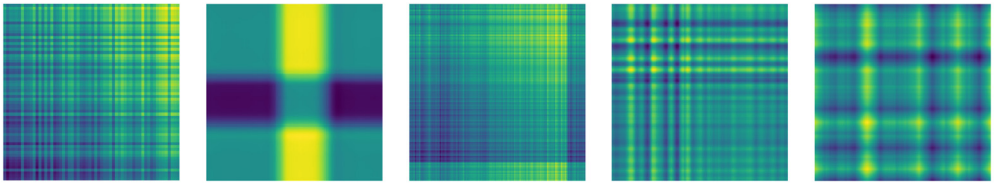



**Fig. 1.** Overall architecture of RPMCNN.



(a) Raw time series data on top five different datasets from the UCR archive: SyntheticControl, GunPoint, CBF, FaceAll, and OSULeaf (from left to right respectively).



(b) Images generated by our framework arrange in the same order with (a).

**Fig. 2.** Images generated by RPMCNN.

as the reference point in each row of $M$, respectively, the transform equation is as follows:

$$M = \begin{bmatrix} x_1 - x_1 & x_2 - x_1 & \cdots & x_m - x_1 \\ x_1 - x_2 & x_2 - x_2 & \cdots & x_m - x_2 \\ \vdots & \ddots & \ddots & \vdots \\ x_1 - x_m & x_2 - x_m & \cdots & x_m - x_m \end{bmatrix} \tag{3}$$

Obviously, every two time stamps of the time series are connected by $M$ to obtain their relative position, each row and column of $M$ contains the information of the whole time series by taking a certain time stamp as a reference point. Furthermore, as one advantage of our work, RPM can be treated as a data augmentation method by offering redundant features of the time series to improve the generalization ability. Each row of $M$ shows the time series with varies reference points and each column shows the mirror of the former, which offers a reversed perspective to view the time series. Finally, min-max normalization is applied to convert $M$ to the gray value matrix, and the final matrix $F$ is obtained by:

$$F = \frac{M - min(M)}{max(M) - min(M)} \times 255 \tag{4}$$

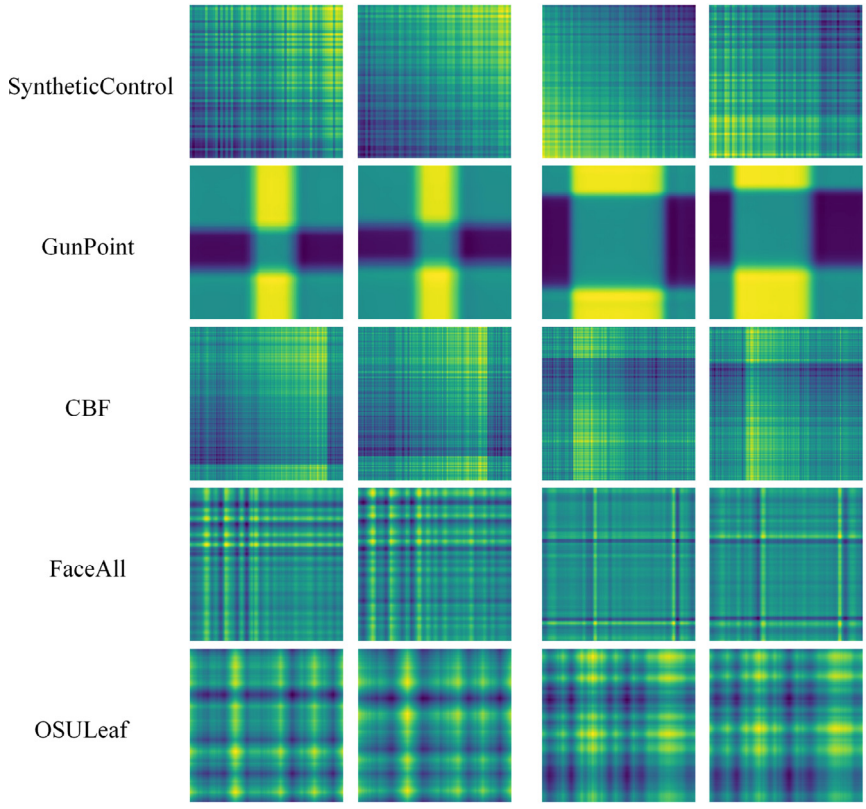
With our proposed pipeline for encoding time series data as 2D images, the converted images can be obtained from the standard datasets such as UCR archive. Fig. 2 shows the raw time series data on top five different datasets from the UCR archive, and images generated by RPM, respectively. The images indicate some useful information of the raw time series data. For instance, the dark area means the lower value of the raw time series and vice versa; the solid color area shows that the value of the raw time series does not change or change slowly; and the frequency of change in color from dark to light or light to dark indicates the waveform of the raw time series, which means high frequency of color changing result in a jagged waveform and low frequency of color changing result in a smoothed waveform. Moreover, the patterns and features embedded in the raw time series data are included in the converted images. As shown in Fig. 3, it is easy to visually see and interpret the similarity of intra-class and inter-class from the images generated by RPM. Then, an improved CNN architecture is applied in the convolution stage to deal with the TSC task based on the 2D images which can also be treated as an image recognition task.

### 3.3. Convolution stage

Once the raw time series data has been converted to images, a CNN model can be trained to classify these images. In this paper, an improved CNN model based on VGGNet is designed. It consists of fewer convolutional layers and fully connected layers to reduce the model complexity, and applies the batch normalization layer to reduce the risk of overfitting, which is more suitable for the circumstance of the TSC task.

As shown in Fig. 4, the symbols “c”, “p”, “f”, “o” in the parentheses refer to the convolutional, pooling, fully connected, and output layers respectively. The symbol D refers to the size of the input image, and N refers to the number of classes of the input images. The numbers before and after “@” refer to the number of feature maps and the shape of a feature map in the present layer respectively. Note that the normalization and rectification layer is not shown in the figure due to the limitation of space.

Our CNN model includes six sections. For the first five sections, each section consists of four layers as follows. The first layer is a convolutional layer which is the core layer of the whole CNN structure. It convolves the output of the previous layer with a set of kernels to extract a set of features at different locations of the original image and obtain the representation which is called the feature maps. The second layer is the batch normalization (BN)




(a) Images of one class on top five different datasets from the UCR archive.

(b) Images of another class on top five different datasets from the UCR archive.

**Fig. 3.** Image similarity of intra-class and inter-class.

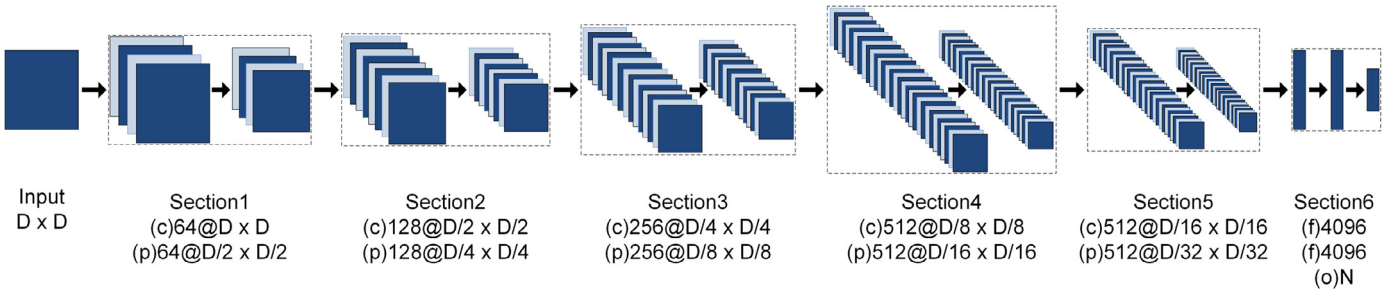



**Fig. 4.** Architecture of RPMCNN.

layer [23] which normalizes the values of different feature maps in the previous layer. It is effective to solve the problem of gradient vanishing and exploding in the process of backpropagation by converting previous unfixed data distributions into fixed data distributions. The third layer is the rectification layer which maps the output of the previous layer by an activation function called SELU [24]. The advantage of SELU is that a self-normalizing attribute is introduced, which enables faster and more robust learning of deep neural networks even in the presence of noise and disturbances. The last layer is the max pooling layer which lowers the computational burden by compressing the feature map processed by previous layers.

The last section consists of three layers including two fully connected layers and an output layer. The role of the fully connected layer is to integrate the features in the feature maps to obtain the high-level denotation of the original image features. The output layer achieves the classification of original images by softmax function which provides the posterior probability of the classification results consistent with the true labels of training samples and predicts the labels of testing samples.

As the whole RPMCNN framework is established, we conduct comprehensive experiments and the results are shown in Section 4.

## 4. Experiments

### 4.1. Experiment settings

We evaluate the performance of all methods on the datasets from the UCR time series classification archive[1]. We select the same 20 datasets as the other state-of-the-art approaches [12,13]. Note that the datasets have already standardized to zero mean and unit variance.

[1] Datasets are available on http://www.cs.ucr.edu/~eamonn/time_series_data/.

**Table 1**
Error rates of different approaches on 20 UCR time series datasets.

| Dataset | 1NN-DTW | BOSS | SAX-VSM | LTS | COTE | FCN | MCNN | GAF-MTF | RPCNN | RPMCNN |
|---|---|---|---|---|---|---|---|---|---|---|
| 50 Words | 0.31 | 0.301 | N/A | 0.232 | 0.191 | 0.321 | **0.19** | 0.301 | 0.26 | 0.193(4) |
| Adiac | 0.396 | 0.22 | 0.38 | 0.437 | 0.233 | **0.143** | 0.231 | 0.373 | 0.28 | 0.182(3) |
| Beef | 0.367 | 0.2 | **0.033** | 0.24 | 0.133 | 0.25 | 0.367 | 0.233 | 0.08 | 0.133(3) |
| CBF | 0.003 | **0** | 0.02 | 0.006 | 0.001 | **0** | 0.002 | 0.009 | 0.005 | **0**(6) |
| Coffee | **0** | **0** | **0** | **0** | **0** | **0** | 0.036 | **0** | **0** | **0**(6) |
| ECG | 0.232 | **0** | 0.14 | **0** | **0** | 0.015 | **0** | 0.09 | **0** | **0**(2) |
| FaceAll | 0.192 | 0.21 | 0.2 | 0.217 | 0.105 | **0.071** | 0.235 | 0.237 | 0.19 | 0.189(2) |
| FaceFour | 0.17 | **0** | **0** | 0.048 | 0.091 | 0.068 | **0** | 0.068 | **0** | 0.034(2) |
| Fish | 0.177 | **0.011** | 0.017 | 0.066 | 0.029 | 0.029 | 0.051 | 0.114 | 0.085 | 0.034(8) |
| GunPoint | 0.093 | **0** | 0.007 | **0** | 0.007 | **0** | **0** | 0.08 | **0** | **0**(1) |
| Lightning2 | 0.131 | 0.148 | 0.19 | 0.177 | 0.164 | 0.197 | 0.164 | 0.114 | **0** | **0**(8) |
| Lightning7 | 0.274 | 0.342 | 0.3 | 0.197 | 0.247 | **0.137** | 0.219 | 0.26 | 0.26 | 0.247(4) |
| OliveOil | 0.167 | **0.1** | **0.1** | 0.56 | **0.1** | 0.167 | 0.133 | 0.2 | 0.11 | **0.1**(16) |
| OSULeaf | 0.409 | **0.012** | 0.107 | 0.182 | 0.145 | **0.012** | 0.271 | 0.358 | 0.29 | 0.186(4) |
| SwedishLeaf | 0.208 | 0.072 | 0.25 | 0.087 | 0.046 | **0.034** | 0.066 | 0.065 | 0.06 | **0.034**(2) |
| SyntheticControl | 0.007 | 0.03 | 0.25 | 0.007 | **0** | 0.01 | 0.003 | 0.007 | **0** | 0.003(1) |
| Trace | **0** | **0** | **0** | **0** | 0.01 | **0** | **0** | **0** | **0** | **0**(4) |
| TwoPatterns | 0.096 | 0.016 | 0.004 | 0.003 | **0** | 0.103 | 0.002 | 0.091 | 0.17 | **0(4)** |
| Wafer | 0.02 | 0.001 | 0.001 | 0.004 | 0.001 | 0.003 | 0.002 | **0** | **0** | **0**(4) |
| Yoga | 0.164 | 0.081 | 0.16 | 0.15 | 0.113 | 0.155 | 0.112 | 0.196 | **0** | 0.11(6) |
| Wins | 2 | 9 | 5 | 4 | 5 | 9 | 5 | 3 | 9 | **10** |
| MR | 7.45 | 3.65 | 5.75 | 5.55 | 3.8 | 4.6 | 4.65 | 6.4 | 3.7 | **2.35** |
| ME | 0.171 | 0.087 | 0.114 | 0.131 | 0.081 | 0.086 | 0.104 | 0.14 | 0.09 | **0.072** |
| MPCE | 0.034 | 0.0137 | 0.0229 | 0.0238 | 0.0152 | 0.0182 | 0.019 | 0.0265 | 0.0108 | **0.0106** |

In our proposed CNN structure, a $3 \times 3$ receptive field with the stride fixed to 1 pixel is applied to each convolutional layer, a $3 \times 3$ pooling window with the stride fixed to 2 pixels is applied to each max pooling layer, and a zero padding technique is applied to remain the size of the feature map unchanged. Moreover, the output nodes of 4096 are applied to each fully connected layer, and a stochastic optimization method called Adagrad [25] is applied to accelerate the training speed.

Our proposed RPMCNN is implemented based on tensorflow and run on NVIDIA GTX 1080Ti graphics cards with 3584 cores and 11 GB global memory. The experiments on each dataset can be split into two phases noted as initial phase and optimization phase. In the initial phase, the hyperparameters of RPMCNN model are set to {$k$, 8, 0.0002} in this research, which denotes the PAA reduction factor, batch size, and learning rate respectively. While the optimization phase which can be described as transfer learning on the same dataset attempts to improve the performance of the previous trained model. In the optimization phase, the learning rate and batch size are fine-tuned using the weight of the previous trained model to escape the previous local optimum to get a better result.

For comparison purposes, we select two baseline methods: 1-NN DTW [26] and FCN [11], one is the baseline of the traditional methods and the other is the baseline of the CNN based methods. Furthermore, we also select 7 state-of-the-art methods within recent 5 years, including BOSS [3], SAX-VSM [27], LTS [28], COTE [4], MCNN [10], GAF-MTF [12], and RPCNN [13]. The experiment results are displayed and discussed in Section 4.3.

### 4.2. Evaluation metrics

In this paper, the performance of all TSC approaches is evaluated by four metrics including the number of datasets on which achieve the lowest error rate (Wins), Mean Rank (MR), Mean Error (ME), and Mean Per-Class Error (MPCE). Specifically, MPCE is defined as the arithmetic mean of the Per-Class Error (PCE). For a dataset pool with $K$ datasets, the $k$th dataset has the number of classes $c_k$ and the corresponding error rate $e_k$, the MPCE is calculated as follows:

$$PCE_k = \frac{e_k}{c_k} \tag{5}$$

$$MPCE = \sum_{k=1}^{K} PCE_k \tag{6}$$

### 4.3. Comparison with state-of-the-art methods

A comprehensive evaluation of our proposed approach and other TSC methods are shown in Table 1. Note that, the numbers in brackets after error rates are the optimal PAA reduction factor of our approach. Since the error rates are missing for some methods on certain datasets, we denote them as âǽN/Aâg and rank them in the last. As shown in Table 1, RPMCNN achieves the best performance in all metrics at first sight, while BOSS, COTE, FCN, MCNN, and RPCNN are also competitive on different metrics.

For intuitive comparison, we group all the approaches into several clusters based on the normalized scores of different metrics (Figs. 5–8). In the metric of Wins, the best group includes RPMCNN, BOSS, FCN, and RPCNN; in the metric of MR, the best group only includes RPMCNN; in the metric of ME, the best group includes RPMCNN, BOSS, FCN, COTE, and RPCNN; and in the metric of MPCE, the best group includes RPMCNN and RPCNN. Furthermore, in each best group, the performance of RPMCNN outperforms the other approaches.

To further evaluate the performance, we make a pairwise comparison for each approach against RPMCNN. We count the number of datasets which performs better, tie, or worse than RPMCNN and apply Binomial Test (BT) and Wilcoxon Rank-sum Test (WRT) to measure the significance of the difference. Corresponding $p$-values are listed in Table 2, which indicates the difference between BOSS, COTE, FCN, RPCNN, and RPMCNN are not significant. It also provides evidence that these approaches achieve better performance than other methods.

For more comparison, we provide more detailed analysis by grouping all the approaches into three categories which denoted as the traditional methods, the 1D-CNN based methods, and the 2D-CNN based methods. Specifically, the 1D-CNN based methods include FCN and MCNN; the 2D-CNN based methods include GAF-MTF and RPCNN; and the rest belongs to the traditional methods.

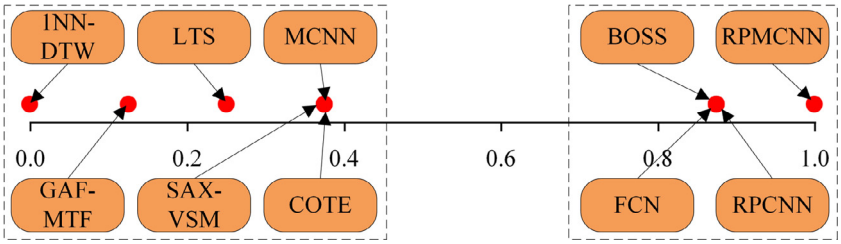



**Fig. 5.** Models grouping by the normalized Wins.

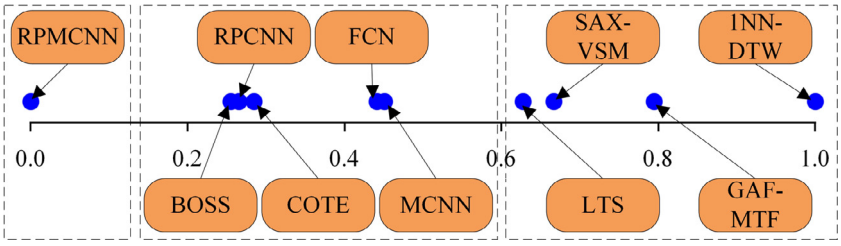



**Fig. 6.** Models grouping by the normalized MR.

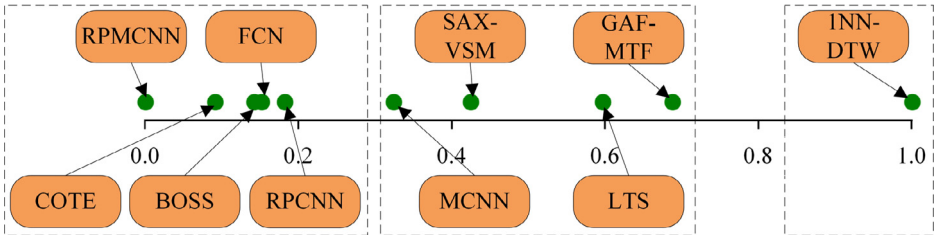



**Fig. 7.** Models grouping by the normalized ME.

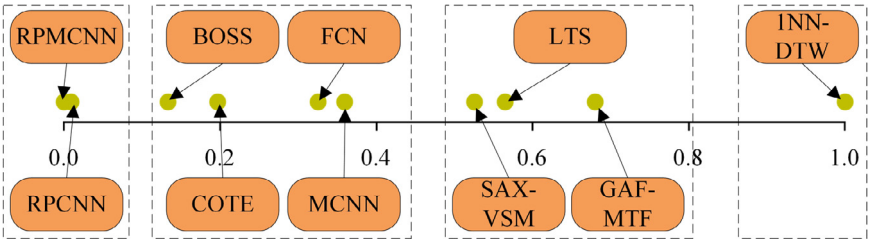



**Fig. 8.** Models grouping by the normalized MPCE.

**Table 2**
Pairwise comparison against RPMCNN.

| Model | better | tie | worse | $p$(BT) | $p$(WRT) |
|---|---|---|---|---|---|
| 1NN-DTW | 0 | 2 | 18 | 7.63E−06 | 1.96E−04 |
| BOSS | 4 | 6 | 10 | 1.79E−01 | 1.77E−01 |
| SAX-VSM | 4 | 3 | 13 | 4.90E−02 | 3.95E−02 |
| LTS | 2 | 4 | 14 | 4.18E−03 | 6.12E−03 |
| COTE | 5 | 6 | 9 | 4.24E−01 | 3.00E−01 |
| FCN | 5 | 5 | 10 | 3.02E−01 | 3.63E−01 |
| MCNN | 3 | 4 | 13 | 2.13E−02 | 1.73E−02 |
| GAF-MTF | 0 | 3 | 17 | 1.53E−05 | 2.92E−04 |
| RPCNN | 4 | 6 | 10 | 1.79E−01 | 1.77E−01 |

In traditional methods, BOSS and COTE are two competitive approaches. However, the performance of other traditional methods are poor with MR above 4, ME above 0.11 and MPCE above 0.022. BOSS uses a distance based on histograms of words to combine the extraction of substructures with the tolerance to extraneous and erroneous data; while COTE utilizes the weighted votes based ensemble structure containing 35 different classifiers constructed in the time, frequency, change, and shapelet transformation domains to obtain higher accuracy. However, there is a huge gap in all metrics compared to RPMCNN.

In 1D-CNN based methods, FCN is claimed to be a standard baseline, which achieves premium performance by simple protocol and small complexity without any crafting in feature engineering and data preprocessing; while MCNN addresses the limitation of extracting features only at a single time scale by performing various transformations of the raw time series data with different downsampling rates and multiple degrees of smoothness. The performance of FCN and MCNN nearly catch up with BOSS and COTE, which demonstrates the feasibility of CNN based methods for the TSC task.

In 2D-CNN based methods, GAF-MTF transforms the raw time series data into a polar coordinate system first and then constructs a Markov transition matrix to convert the 1D time series data to 2D images; while RPCNN applies a method called recurrence plots to reveal in which points some trajectories return to a previous state. However, RPMCNN outperforms both of them on each metric, which proves the effectiveness of the combination of RPM and CNN. To figure out where the performance improvement comes from, we conduct more ablation studies in Section 4.4–4.6.

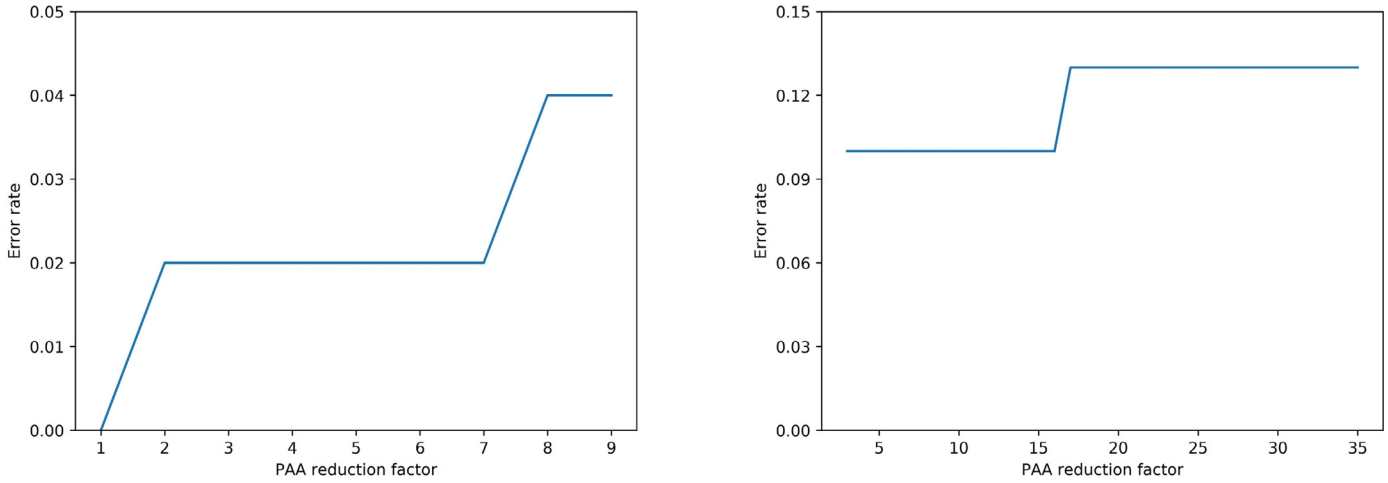



**Fig. 9.** Effect of PAA reduction factor on different datasets.

### 4.4. Comparison with different PAA reduction factor

Since PAA is applied for dimensionality reduction in this paper, we study the effect of the PAA reduction factor. Note that we choose GunPoint and OliveOil datasets to evaluate the performance with different PAA reduction factor since the two datasets have the minimum and maximum optimal PAA reduction factor, respectively, within the 20 datasets, and the batch size and learning rate of RPMCNN are the same depicted in Section 4.1. In order to ensure the image quality and the processing time, we make sure that the size of the converted image is no less than 16x16 and no more than $256 \times 256$. As the length of GunPoint and OliveOil dataset is 150 and 570, we set the range of the PAA reduction factor to 1–9 and 3–35, respectively. As shown in Fig. 9, the long-term trend of the error rate rises with the increase of the PAA reduction factor and the short-term trend is stable. Since the PAA reduction factor gets larger, the time series gets more smooth. However, excessive smoothness will lead to feature loss. Therefore, the error rate rises when the PAA reduction factor exceeds a certain threshold, which means too much dimensionality reduction. On the other hand, the degree of dimensionality reduction seriously affects the memory and time consumption. Specifically, small PAA reduction factor results in large image size which consumes larger memory and time, and vice versa. Therefore, we choose the largest PAA reduction factor leading to the lowest error rate as the optimal one.

### 4.5. Comparison with other 2D representations

Since RPMCNN is also a 2D-CNN based method, we provide different representation methods including GAF-MTF and RP at the representation stage to compare with RPM. Note that the batch size and learning rate of the CNN model are the same depicted in Section 4.1. Since GAF-MTF also uses PAA to smooth the time series, we set the PAA reduction factor the same depicted in Table 1. Moreover, according to the experiment setup of RPCNN [13], we rescale the size of images converted by RP to $64 \times 64$. As shown in Table 3, RPM achieves better results than other representation methods.

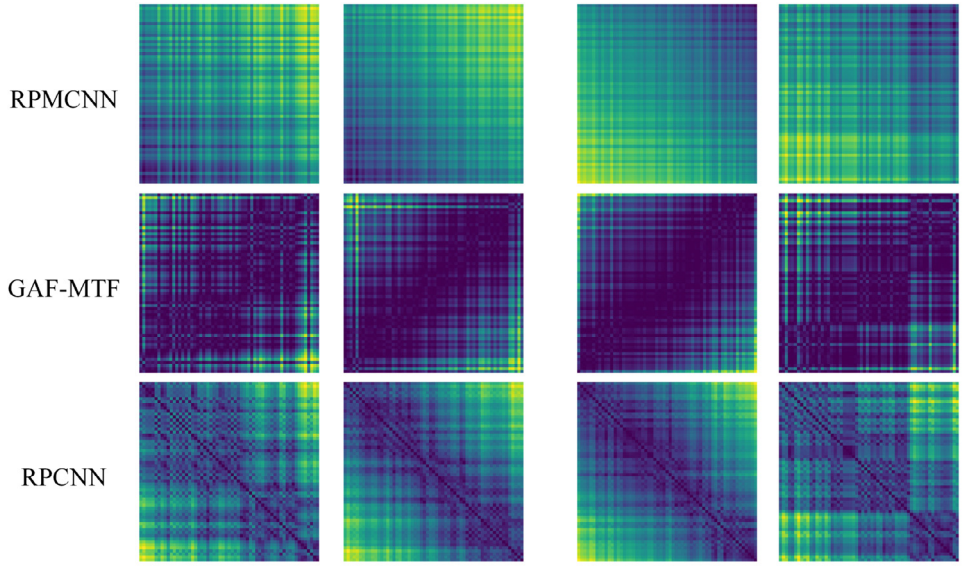
To figure out why RPM outperforms other representation methods, we show the images converted by different representation approaches in Fig. 10. We choose two different classes of two datasets and visually see the intra-class similarity and inter-class similarity. We find that RPMCNN performs well on both two datasets since it achieves high intra-class similarity and low inter-class similarity, while GAF-MTF and RPCNN perform poorly on the SyntheticControl dataset. To obtain quantitative results of the similarity between two converted images, we measure the similarity based on the Euclidean distance which is calculated as follows:

$$L_2(A, B) = \left[ \sum_{i=1}^{n} (a_i - b_i)^2 \right]^{1/2} \tag{7}$$

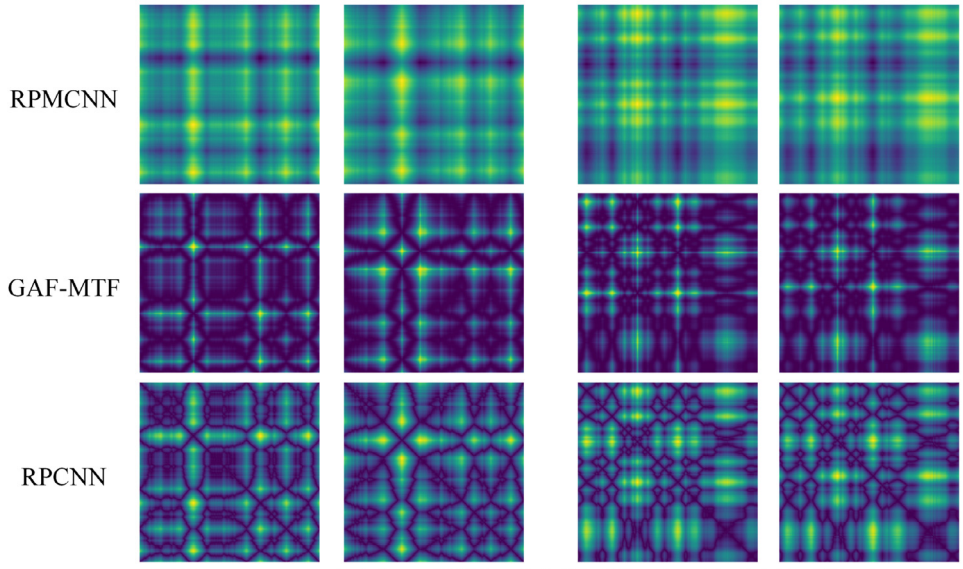
where $A$ and $B$ represent two images, $a_i$ and $b_i$ is the normalized pixel value of the two images respectively, $n$ is the number of pixels. Therefore, the higher distance indicates the lower similarity and vice versa. We randomly choose 4 images from each class and measure the average intra-class similarity and inter-class similarity, which is shown in Fig. 11. Note that the blue bar and red bar in the figure represents the Euclidean distance of intra-class and inter-class, respectively. The results show that the images converted by RPMCNN achieve low intra-class distance and high inter-class dis-

**Table 3**
Results of other representations with our CNN structure.

| Dataset | GAF-MTF | RP | RPM |
|---|---|---|---|
| 50Words | 0.204 | 0.237 | **0.193** |
| Adiac | 0.279 | 0.217 | **0.182** |
| Beef | 0.3 | 0.3 | **0.133** |
| CBF | 0.003 | 0.006 | **0** |
| Coffee | **0** | **0** | **0** |
| ECG | **0** | **0** | **0** |
| FaceAll | 0.269 | 0.239 | **0.189** |
| FaceFour | 0.091 | 0.057 | **0.034** |
| Fish | 0.097 | 0.051 | **0.034** |
| GunPoint | 0.013 | 0.027 | **0** |
| Lightning2 | **0** | **0** | **0** |
| Lightning7 | 0.274 | 0.315 | **0.247** |
| OliveOil | **0.1** | 0.133 | **0.1** |
| OSULeaf | 0.285 | 0.24 | **0.186** |
| SwedishLeaf | 0.061 | 0.054 | **0.034** |
| SyntheticControl | 0.297 | 0.33 | **0.003** |
| Trace | **0** | **0** | **0** |
| TwoPatterns | 0.49 | 0.489 | **0** |
| Wafer | **0** | **0** | **0** |
| Yoga | 0.161 | 0.129 | **0.11** |
| Wins | 6 | 5 | **20** |
| MR | 2.1 | 2.05 | **1** |
| ME | 0.146 | 0.141 | **0.072** |
| MPCE | 0.0252 | 0.0244 | **0.0106** |




(a) Images converted from SyntheticControl, left half is class 3, right half is class 4.




(b) Images converted from OSULeaf, left half is class 3, right half is class 4.

**Fig. 10.** Images converted by 2D-CNN based methods.

tance on both two datasets, which means high intra-class similarity and low inter-class similarity. In this case, it is easier for the following step of recognition. However, GAF-MTF and RPCNN perform poorly on the OSULeaf dataset as the Euclidean distance of intra-class and inter-class are nearly the same. Since all methods perform well on the OSULeaf dataset, we mainly focus on the SyntheticControl dataset to find out the difference between RPMCNN and the other methods. The SyntheticControl dataset is artificially synthesized, the trends of its two classes are upward and downward respectively, which is almost the mirror of each other except for the random noise as is shown in Fig. 12. Meanwhile, the images converted by GAF-MTF and RPCNN are diagonally symmetric from top left to bottom right, which results in the same image converted from the original time series data and its mirror. Therefore, it is hard to distinguish the images of the two classes converted by GAF-MTF and RPCNN on the SyntheticControl dataset. However, RPMCNN offers a reversed perspective to view the time series by building a matrix of which the elements on different sides of the diagonal are opposite to each other, which is able to deal with the situation of data mirroring. Therefore, RPMCNN still performs well on the SyntheticControl dataset, and the characteristic of converting time series to highly recognizable images may explain the remarkable performance of RPMCNN.

### *4.6. Comparison with other CNN structures*

To validate the effectiveness of our proposed CNN structure, we first carry out comparing experiments on the models with/without BN layers to find out how BN layers affect the performance. Then we use different CNN structures including the classic VGGNet-13, VGGNet-16, ResNet-50, and ResNet-101 at the convolution stage to compare with RPMCNN, which is shown in Table 4. Note that each convolutional layer is followed by BN, and the hyperparameters of all CNN models are the same depicted in Section 4.1.

The results show that adding BN layers can improve the performance since it enables more effective and stable optimization by reparametrization. Specifically, the smoothing effect of reparametrization makes the training process converging to more

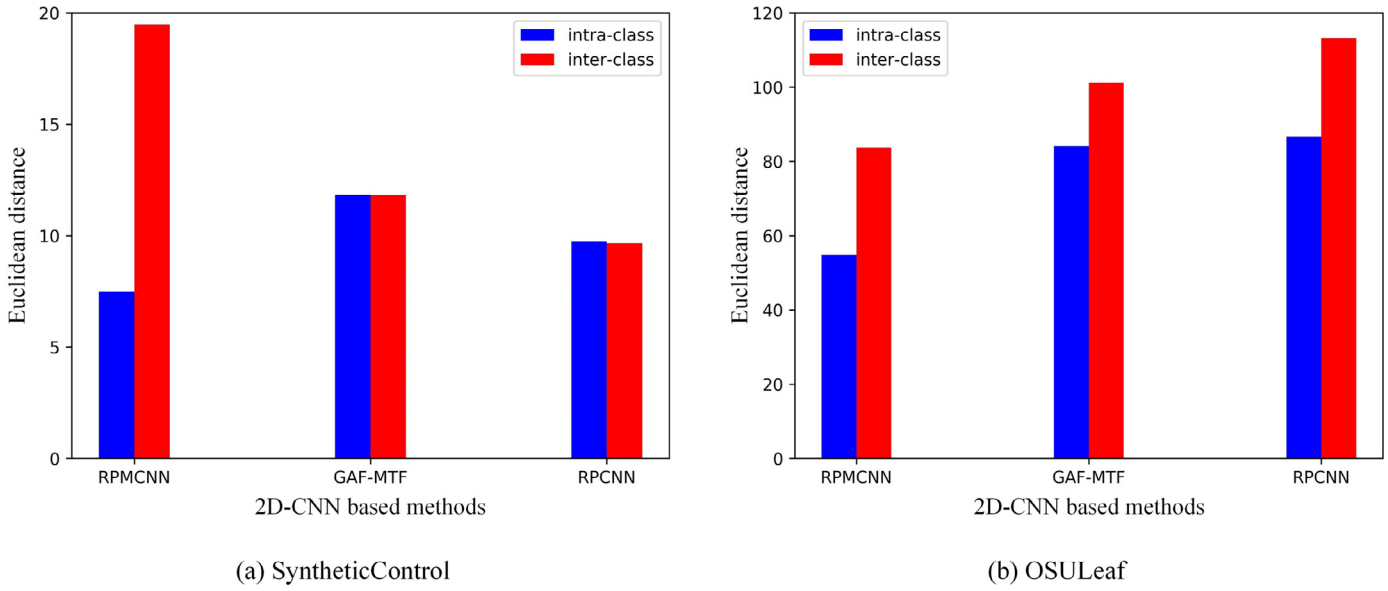



**Fig. 11.** Similarity measurement of images converted from different datasets.

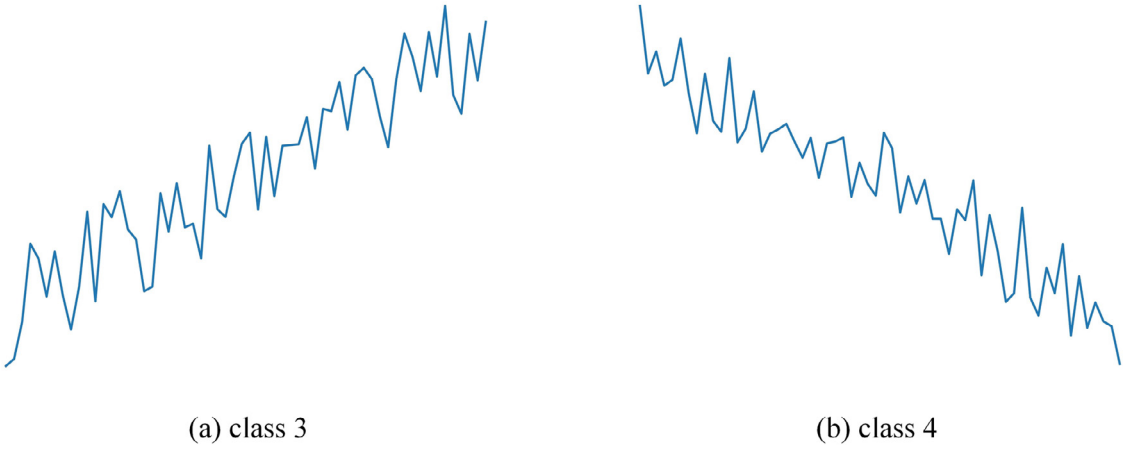



**Fig. 12.** Trends of SyntheticControl dataset.

**Table 4**
Results of other CNN structures with RPM.

| Dataset | VGGNet-13 | VGGNet-16 | ResNet-50 | ResNet-101 | RPMCNN (without BN) | RPMCNN |
|---|---|---|---|---|---|---|
| 50Words | 0.209 | 0.226 | 0.259 | 0.257 | 0.231 | **0.193** |
| Adiac | 0.2 | 0.2 | 0.235 | 0.253 | 0.312 | **0.182** |
| Beef | 0.2 | 0.2 | 0.2 | 0.2 | 0.267 | **0.133** |
| CBF | **0** | **0** | 0.013 | 0.02 | 0.001 | **0** |
| Coffee | **0** | **0** | **0** | **0** | **0** | **0** |
| ECG | **0** | **0** | **0** | **0** | **0** | **0** |
| FaceAll | **0.189** | 0.193 | 0.225 | 0.258 | 0.268 | **0.189** |
| FaceFour | 0.068 | 0.068 | 0.068 | **0.034** | 0.091 | **0.034** |
| Fish | **0.034** | **0.034** | 0.126 | 0.109 | 0.12 | **0.034** |
| GunPoint | 0.027 | 0.04 | 0.053 | 0.067 | 0.053 | **0** |
| Lightning2 | **0** | **0** | **0** | **0** | **0** | **0** |
| Lightning7 | 0.315 | 0.315 | 0.329 | 0.342 | 0.315 | **0.247** |
| OliveOil | **0.1** | 0.133 | **0.1** | **0.1** | 0.233 | **0.1** |
| OSULeaf | 0.207 | 0.236 | 0.351 | 0.397 | 0.302 | **0.186** |
| SwedishLeaf | 0.038 | 0.043 | 0.066 | 0.07 | 0.074 | **0.034** |
| SyntheticControl | **0.003** | **0.003** | 0.01 | 0.007 | **0.003** | **0.003** |
| Trace | **0** | **0** | **0** | **0** | **0** | **0** |
| TwoPatterns | **0** | **0** | **0** | **0** | **0** | **0** |
| Wafer | **0** | **0** | **0** | **0** | **0** | **0** |
| Yoga | **0.11** | 0.112 | 0.164 | 0.149 | 0.162 | **0.11** |
| Wins | 12 | 9 | 7 | 8 | 7 | **20** |
| MR | 1.45 | 2.05 | 3.35 | 3.35 | 3.55 | **1** |
| ME | 0.085 | 0.09 | 0.11 | 0.113 | 0.122 | **0.072** |
| MPCE | 0.013 | 0.0141 | 0.0176 | 0.0178 | 0.0196 | **0.0106** |

flat minima, which is proved by Santurkar et al. [29]. This may help understanding the wide use of BN layers in deep learning. Therefore, we apply the BN layers to other CNN structures to achieve better performance for comparison.

The results also demonstrate that RPMCNN is more effective than other CNN structures. Meanwhile, we find that the ME and MPCE increase as the CNN structure gets deeper. Generally, the deeper structure of CNN achieves better results in recent machine learning competitions since it may learn some highly complex patterns in the data. However, it requires more effort in preventing overfitting to improve the model generalization ability. Since the patterns in the time series data provided by UCR archive are not so complex to catch comparing with the real world images, VGGNet and ResNet which are specifically designed for image analysis tend to overfit easier. Moreover, the research carried by Wang et al. [11] also comes to the same conclusion, which helps to explain why the performance of deeper structures is not as good as ours. Therefore, as an appropriate structure we proposed, RPMCNN is more suitable for the TSC task.

With the comprehensive comparison above, we have validated the availability of CNN based methods. More importantly, as a novel CNN based framework, RPMCNN achieves remarkable performance in all aspects and proves that the CNN based method is more effective and adaptive than the traditional TSC approach.

## 5. Conclusions and future work

We have proposed a novel framework RPMCNN for the TSC task, using an improved CNN architecture to recognize the 2D images converted from time series data by RPM. RPM contains redundant features of the raw time series and makes it easier to see and interpret the similarity of inter-class and intra-class from the converted images, and the improved CNN architecture is constructed with fewer layers to avoid overfitting and boost the generalization ability. We have conducted comprehensive experiments and compared with competitive TSC methods in recent years. The experimental results prove that RPMCNN achieves state-of-the-art performance and outperforms the existing methods by a large margin. We have also carried out ablation studies of 2D representations and CNN structures respectively, which proves the combination of RPM and our CNN structure is more suitable for the TSC task.

In future work, we will explore deeper structures to process larger and more complex time series data from the real world applications, and apply our approach in more difficult TSC tasks such as multivariate time series scenario. Another important future work we are interested in is how different deep learning architectures perform on the TSC task and exploring the ensemble learning method of different architectures to achieve better performance.

## Declaration of competing interest

We declare that all authors have seen and approved the final version of the manuscript being submitted. We warrant that the article is the original work which has not been published previously and is not currently submitted for review to any other journal.

We wish to draw the attention of the Editor to the following facts which may be considered as potential conflicts of interest and to significant financial contributions to this work. We confirm that the manuscript has been read and approved by all named authors and that there are no other persons who satisfied the criteria for authorship but are not listed. We further confirm that the order of authors listed in the manuscript has been approved by all of us.

We confirm that we have given due consideration to the protection of intellectual property associated with this work and that there are no impediments to publication, including the timing of publication, with respect to intellectual property. In so doing we confirm that we have followed the regulations of our institutions concerning intellectual property.

We understand that the Corresponding Author is the sole contact for the Editorial process (including Editorial Manager and direct communications with the office). He is responsible for communicating with the other authors about progress, submissions of revisions and final approval of proofs. We confirm that we have provided a current, correct email address which is accessible by the Corresponding Author and which has been configured to accept email from keshi@mail.hust.edu.cn.

## Acknowledgments

This research was supported by the Natural Science Foundation of China under Grant no. 51435009.

## Supplementary material

Supplementary material associated with this article can be found, in the online version, at doi:10.1016/j.neucom.2019.06.032.

## References

[1] J. Paparrizos, L. Gravano, Fast and accurate time-series clustering, ACM Trans. Datab. Syst. 42 (2) (2017) 1–49.
[2] 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 Mining Knowl. Discov. 31 (3) (2017) 606–660.
[3] P. Schfer, The boss is concerned with time series classification in the presence of noise, Data Min. Knowl. Discov. 29 (6) (2015) 1505–1530.
[4] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with cote: the collective of transformation-based ensembles, IEEE Trans. Knowl. Data Eng. 27 (9) (2015) 2522–2535.
[5] A. Karpathy, G. Toderici, S. Shetty, T. Leung, R. Sukthankar, F.F. Li, Large-scale video classification with convolutional neural networks, in: Proceedings of the 2014 IEEE Conference on Computer Vision and Pattern Recognition, 2014, pp. 1725–1732.
[6] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436–444.
[7] Y. Gong, Q. Zhang, Hashtag recommendation using attention-based convolutional neural network, in: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, 2016, pp. 2782–2788.
[8] A. Severyn, A. Moschitti, Learning to rank short text pairs with convolutional deep neural networks, in: Proceedings of the Thirty-Eighth International ACM SIGIR Conference on Research and Development in Information Retrieval, 2015, pp. 373–382.
[9] Y. LeCun, Y. Bengio, in: Convolutional networks for images, speech, and time series, 1995, p. 3361.
[10] Z. Cui, W. Chen, Y. Chen, Multi-scale convolutional neural networks for time series classification, 2016, http://arxiv.org/abs/1603.06995.
[11] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: a strong baseline, in: Proceedings of the 2017 International Joint Conference on Neural Networks, 2017, pp. 1578–1585.
[12] Z. Wang, T. Oates, Imaging time-series to improve classification and imputation, in: Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015, pp. 3939–3945.
[13] N. Hatami, Y. Gavet, J. Debayle, Classification of time-series images using deep convolutional neural networks, 2017, http://arxiv.org/abs/1710.00886.
[14] D.E. Rumelhart, J.L. McClelland, Learning Internal Representations by Error Propagation, MIT Press, pp. 318–362.
[15] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the 2016 IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.
[16] Q.V. Le, J. Ngiam, Z. Chen, D. Chia, P.W. Koh, A.Y. Ng, Tiled convolutional neural networks, in: Proceedings of the Twenty-Third Annual Conference on Neural Information Processing Systems, 2010, pp. 1279–1287.
[17] J.P. Eckmann, S.O. Kamphorst, D. Ruelle, Recurrence plots of dynamical systems, Europhys. Lett. 4 (9) (1987) 973–977.
[18] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, in: Proceedings of the Twenty-Fifth Annual Conference on Neural Information Processing Systems, 2012, pp. 1106–1114.
[19] K. Simonyan, A. Zisserman, Very deep convolutional networks for large-scale image recognition, 2015, http://arxiv.org/abs/1409.1556.
[20] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the 2015 IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[21] Y. Lecun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[22] E.J. Keogh, M.J. Pazzani, A simple dimensionality reduction technique for fast similarity search in large time series databases, in: Proceedings of the Fourth Pacific-Asia Conference on Knowledge Discovery and Data Mining, 2000, pp. 122–133.
[23] S. Ioffe, C. Szegedy, Batch normalization: accelerating deep network training by reducing internal covariate shift, in: Proceedings of the Thirty-Second International Conference on Machine Learning, 2015, pp. 448–456.
[24] G. Klambauer, T. Unterthiner, A. Mayr, S. Hochreiter, Self-normalizing neural networks, 2017, arXiv:1706.02515.
[25] J.C. Duchi, E. Hazan, Y. Singer, Adaptive subgradient methods for online learning and stochastic optimization, J. Mach. Learn. Res. 12 (2011) 2121–2159.
[26] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: Proceedings of the Workshop on Knowledge Discovery in Databases, 1994, pp. 229–248.
[27] P. Senin, S. Malinchik, Sax-VSM: interpretable time series classification using sax and vector space model, in: Proceedings of the 2013 IEEE International Conference on Data Mining, 2013, pp. 1175–1180.
[28] J. Grabocka, N. Schilling, M. Wistuba, L. Schmidt-Thieme, Learning time-series shapelets, in: Proceedings of the Twentieth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 392–401.
[29] S. Santurkar, D. Tsipras, A. Ilyas, A. Madry, How does batch normalization help optimization? 2018, arXiv:1805.11604.

**Wei Chen** received the B.S. degree in measurement and control technology and instrumentation, and the M.E. degree in control engineering from Huazhong University of Science and Technology, Wuhan, Hubei, PR China. He is currently studying toward the Ph.D. degree in the School of Computer Science and Technology, Huazhong University of Science and Technology. His main research interest is deep learning, time series analytics and its applications.

**Prof. Ke Shi** received his B.S. and Ph.D. degree from Huazhong University of Science and Technology in 1994 and 2000, respectively. From 2004 to 2006, he worked in Wireless network Lab at Cornell University as a visiting associate professor. Since 2008, he has been a full professor with the School of Computer Science and Technology, Huazhong University of Science and Technology. His current research interests include ad hoc and wireless networks, deep learning, big data analytics and its industrial applications.