Neural Networks 136 (2021) 126–140

Contents lists available at ScienceDirect

Neural Networks

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

Multi-scale Attention Convolutional Neural Network for time series classification[☆]

Wei Chen, Ke Shi [*]

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

ARTICLE INFO

Article history:
Received 10 September 2019
Received in revised form 5 November 2020
Accepted 3 January 2021
Available online 6 January 2021

Keywords:
Time series classification
Convolutional neural network
Multi-scale attention mechanism

ABSTRACT

With the rapid increase of data availability, time series classification (TSC) has arisen in a wide range of fields and drawn great attention of researchers. Recently, hundreds of TSC approaches have been developed, which can be classified into two categories: traditional and deep learning based TSC methods. However, it remains challenging to improve accuracy and model generalization ability. Therefore, we investigate a novel end-to-end model based on deep learning named as Multi-scale Attention Convolutional Neural Network (MACNN) to solve the TSC problem. We first apply the multi-scale convolution to capture different scales of information along the time axis by generating different scales of feature maps. Then an attention mechanism is proposed to enhance useful feature maps and suppress less useful ones by learning the importance of each feature map automatically. MACNN addresses the limitation of single-scale convolution and equal weight feature maps. We conduct a comprehensive evaluation of 85 UCR standard datasets and the experimental results show that our proposed approach achieves the best performance and outperforms the other traditional and deep learning based methods by a large margin.

1. Introduction

As we enter a new era of Internet of Things (IoT), a large amount of time series data is exploding in various fields including engineering, finance, medicine, and others (Paparrizos & Gravano, 2017). Time series data mining has become one of the top ten most challenging research issues in the field of data mining at the 21st century. Among all tasks of time series data mining, time series classification (TSC) always has attracted widespread attention from academia and industry.

The goal of TSC task is to build a classifier that can predict a class label according to an ordered set of real-valued attributes. In the last two decades, a large amount of TSC methods have been proposed, most of them are traditional methods and can be divided into three types: distance-based, feature-based, and ensemble-based.

For distance-based approaches, the regular way is to perform pre-defined similarity measurements directly on raw time series and then use the nearest neighbor (NN) classifier. Conventional methods for similarity measurement include Euclidean Distance (ED) and Dynamic Time Warping (DTW). However, more and more researchers believe that in most domains, DTW is more effective as a tool of similarity measurement than ED (Xi et al., 2006). Therefore, DTW combined with NN classifier is the most notable distance-based method, which is also a very strong baseline (Bagnall et al., 2017).

For feature-based approaches, the most important part is to find the patterns of time series by extracting a set of features. To name several representative approaches, Time Series Bag of Features (TSBF) (Baydogan et al., 2013) extracts the interval features by selecting random subseries to find summary statistics, then apply random forest classifier to solve the classification task. Bag of SFA Symbols (BOSS) (Schäfer, 2015) is built on Symbolic Fourier Approximation (SFA) (Schäfer & Högqvist, 2012) to form words histogram according to the words distribution, and then the classification result is obtained by the distance of words histogram. Shapelet Transform (ST) (Hills et al., 2014) proposes a single-scan algorithm using shapelet to produce a transformed dataset, a variety of traditional classifiers can be selected to classify the transformed data, which is more flexible and effective than the tree-based shapelet classifier first proposed by Ye and Keogh (Ye & Keogh, 2011).

For ensemble-based approaches, different classifiers are combined to achieve higher accuracy. Elastic Ensemble (PROP) (Lines & Bagnall, 2015) constructs 11 sub-classifiers by combining 11 elastic distance measures with 1NN classifier, then a weighted voting scheme is applied to the sub-classifiers to get the classification result. The Flat Collective of Transformation Ensembles

[☆] This document is the results of the research project funded by the National Natural Science Foundation of China.

[*] 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.neunet.2021.01.001
0893-6080/© 2021 Elsevier Ltd. All rights reserved.

(Flat-COTE) (Bagnall et al., 2015) employs 35 classifiers constructed in the time and frequency domains into a flat hierarchy to achieve better performance. Later, the Hierarchical Vote Collective of Transformation-based Ensembles (Hive-COTE) (Lines et al., 2018) offers a new hierarchical structure with probabilistic voting to overcome the deficiencies of Flat-COTE.

The approaches mentioned above have achieved remarkable performance. However, with the development of artificial intelligence, deep learning based TSC approaches have emerged rapidly and shown competing performance until recent years. Inspired by the achievements of deep leaning based technologies, we propose a novel deep learning architecture named as Multi-scale Attention Convolutional Neural Network (MACNN) to improve the performance of TSC task. Specifically, stacked Multi-scale Attention (MA) module is developed to implement the Multi-scale Attention Mechanism (MAM) which is a strategy that enhances useful feature maps and suppresses less useful ones by learning the importance of each feature map automatically. The MA module consists of a multi-scale block and an attention block. The multi-scale block which produces different sizes of the receptive field to capture different scales of information is built up of concatenated convolutional layers with different kernel size. The attention block brings in a channel-wise weight to focus on the important feature maps by squeezing and rescaling. Therefore, MACNN is a powerful end-to-end model without heavy crafting in data pre-processing. We conduct a comprehensive evaluation of 85 datasets from the UCR time series classification archive (Bagnall et al., 2018). The experimental results demonstrate that our proposed method outperforms the other approaches by a large margin.

The main contributions of this paper can be summarized as follows. Firstly, the MAM is proposed to improve accuracy and model generalization ability using multi-scale convolution and attention mechanism. Secondly, the MACNN architecture is built on stacked MA modules which consist of the multi-scale block and attention block. The multi-scale block is proposed to obtain different scales of information and the attention block is developed on channel dimension to focus on the important feature maps. Thirdly, sufficient experiments have been done and the results are evaluated with different metrics, which proves that our proposed method achieves better performance than other methods. Finally, the use of Class Activation Map (CAM) is investigated to figure out the effect of MAM and explain the recognition ability of a network.

The rest of this paper is organized as follows. We introduce the definition of TSC task and related work in Section 2. We describe the architecture of our proposed model in Section 3. We present the classification and visualization results of UCR standard datasets in Section 4. At last, the conclusions and future work are provided in Section 5.

2. Background and related work

2.1. Time series classification

In this section, we will present two definitions for TSC task before introducing the deep learning based TSC approaches.

Definition 1. A univariate time series $T = [x_1, x_2, \ldots, x_L]$ is a sequence of real-valued data points with timestamps. Noted that L is the length of T.

Definition 2. A dataset $D = \{(T_1, Y_1), (T_2, Y_2), \ldots, (T_M, Y_M)\}$ contains M pairs of (T_i, Y_i), where (T_i, Y_i) is the ith sample and Y_i is the corresponding one-hot label vector of T_i.

The TSC task is to build a classification model that maps from the time series to the most possible class label.

2.2. Deep learning for TSC

Deep learning has achieved remarkable progress in recent years and spread rapidly to many domains, which leads researchers to explore deep learning based methods to solve sophisticated problems in TSC field.

The deep learning based TSC methods can be divided into two categories: the generative and discriminative methods (Längkvist et al., 2014). The generative methods which can be treated as model-based methods in the TSC community (Bagnall et al., 2017) are aimed at finding an appropriate time series representation before training a classifier. Hu et al. (2016) proposed a stacked denoising auto-encoder (SDAE) for the pre-training phase. Serrà et al. (2018) developed a universal neural network encoder to transform variable-length time series into a fixed-length, low-dimensional representation. Moreover, Deep Belief Network (DBN) was used in an unsupervised way to model the latent features of time series (Banerjee et al., 2017). Echo State Network (ESN) was also used to learn the appropriate representation by reconstructing the raw time series (Aswolinskiy et al., 2017). Since an additional step of unsupervised training is introduced, the implementation of generative methods is often more complicated. Furthermore, the performance of generative methods is usually worse than discriminative methods since the latter directly learn the mapping between the raw time series and the class probability distribution. Therefore, these barriers lead researchers to focus on discriminative methods.

Among discriminative methods, we mainly focus on the end-to-end approach since the raw time series is fed to deep learning model without heavy crafting in data pre-processing compared with feature engineering based methods such as Gramian fields and Markov transition fields (Wang & Oates, 2015). To name a few attracting end-to-end approaches, Wang et al. (2017) applied three deep learning models including Multilayer Perceptron (MLP) (Rumelhart & McClelland, 1987), Fully Convolutional Networks (FCN) (Long et al., 2015), and ResNet (He et al., 2016) to TSC, which provides a strong baseline for further work. Recently, Fawaz et al. (2019) proposed a comprehensive review of deep learning based TSC methods. Their research indicates that CNN is the most commonly used structure for TSC task, which may due to the robustness and less training time compared to other deep learning architectures such as Recurrent Neural Network (RNN) or MLP. More importantly, they chose nine deep learning based approaches including MLP, CNN, FCN, ResNet, and ESN, then evaluated the performance of each approach on the UCR datasets and came to the conclusion that ResNet and FCN are the best two methods.

In summary, the researches mentioned above apply various kinds of deep learning architectures to deal with TSC task. However, except for the fine-tuning of the parameters, the architectures are usually off-the-shelf. Therefore, we intend to investigate a new CNN architecture for TSC task to achieve better performance.

2.3. Attention mechanism

An attention mechanism is a tool to bias the allocation of available processing resources towards the most important features of the input data. The goal of the attention mechanism is to enhance the representation ability of a network by focusing on important features and suppressing unnecessary ones. Recently, how to use the attention mechanism in CNN has become a research hotspot.

There are several typical attention models such as Spatial Transformer Networks (STN) (Jaderberg et al., 2015), Squeeze-and-Excitation (SE) Networks (Hu et al., 2019), and Convolutional Block Attention Module (CBAM) (Woo et al., 2018). STN provides

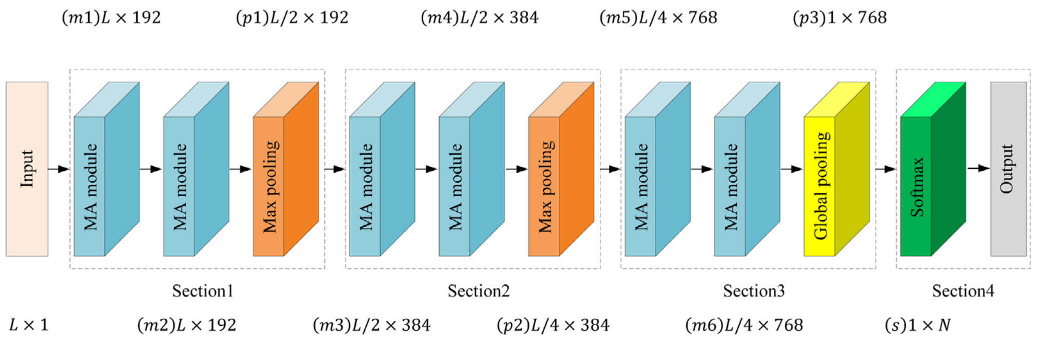


Fig. 1. Overall architecture of MACNN.

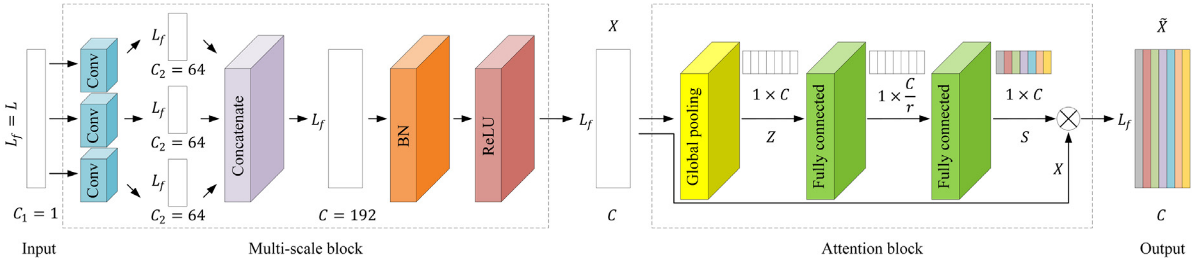


Fig. 2. Structure of the MA module.

spatial transformation capabilities by producing an appropriate transformation for each input sample. The transformation which can include scaling, cropping, rotations, and non-rigid deformations is then performed on the entire feature map. This allows networks to select the most relevant regions of an image and transform those regions into a canonical pose to simplify recognition in the following layers. Therefore, STN can be treated as spatial attention. While Hu et al. propose a SE block that focuses on channel attention. The SE block enables the whole network to use global information to selectively focus on the important feature maps and suppress less important ones, which is called feature recalibration. More importantly, the SE block enhances the quality of the shared lower-level representations in the early layers and becomes increasingly specialized when responding to different inputs in later layers. Therefore, the weight of each feature map is automatically learned at each layer of the network and the benefits of the SE block can be accumulated through the entire network to boost the feature discriminability. Furthermore, Woo et al. propose a CBAM which sequentially applies channel and spatial attention modules to emphasize meaningful features along the channel and spatial axes.

Since time series data has no spatial axes, which is different from image data, we intend to implement attention mechanism on channel dimension to improve the accuracy of classification.

3. Methodology

3.1. MACNN architecture

The overall architecture of MACNN is depicted in Fig. 1. The symbols "m", "p", and "s" refer to the MA, pooling and softmax layer respectively. The symbol "L" and "N" refer to the time series length and the number of classes respectively. The output of each layer is denoted in Fig. 1. Noted that, the numbers before and after "×" refer to the length and channels of feature maps respectively.

There are four sections in MACNN architecture. For the first two sections, each section consists of the following three layers. The first two layers are the MA layer which is the core layer of the whole architecture. The MA layer includes an MA module that implements the MAM to learn the importance of each feature map generated by the multi-scale convolution. The structure of the MA module is displayed and discussed in Section 3.2. The last layer is the max-pooling layer which reduces the number of parameters to control overfitting and to improve model performance by selecting the features with stronger identification.

The structure of the third section is almost the same as the previous sections except for the pooling layer. In this section, we apply a global average pooling layer (Lin et al., 2014) instead of the fully connected layer to generate the final feature maps for classification. The advantage of the global average pooling is that it can largely decrease the number of parameters to reduce the risk of overfitting.

The last section is the output section which gives the classification result by a softmax layer. The softmax layer provides the posterior probability of each class which is coded as one-hot format and passed to the output.

3.2. Multi-scale attention mechanism

The MAM is a strategy that enhances useful feature maps and suppresses less useful ones according to the importance of each feature map generated by the multi-scale convolution. The goal of the MAM is to improve the recognition ability of a network. To achieve this goal, the multi-scale convolution is proposed first to obtain multi-scale temporal information which refers to the short-term, mid-term, and long-term dependencies of time series. Then the attention mechanism is applied to selectively focus on the important information and ignore the unimportant ones by rescaling the weight of each feature map. Therefore, MAM is expected to improve classification performance.

The MAM is implemented by the MA module which is the core module of the MACNN architecture. The structure of MA module consists of a multi-scale block and an attention block. For better understanding the structure of MA module, we take the first layer of MACNN as an example which is shown in Fig. 2. The symbols "C_1", "L_f", and "C_2" refer to the channels of input feature maps, the length of input feature maps, and the output channels of each convolutional layer in the multi-scale block respectively. Let $X \in \mathbb{R}^{L_f \times C}$ and $\tilde{X} \in \mathbb{R}^{L_f \times C}$ be the output of the multi-scale block and the attention block respectively, where C refers to the number of channels and noted that $C = 3C_2$ since there are 3 convolutional layers in the multi-scale block. The multi-scale block is built up of concatenated convolutional layers with different kernel sizes, which is enlightened by the structure of inception module (Szegedy et al., 2015). The multi-scale block produces different sizes of the receptive field to capture different scales of temporal information; The attention block brings in a channel-wise weight to focus on the important feature maps, which enhances the recognition ability for classification. Noted that, since the input is the raw time series, the channels of input feature maps are 1 and the length of input feature maps is L. Moreover, the number of filters is set to 64 in the convolutional layer of the first MA module. Therefore, the output channels of each convolutional layer are 64. More details are given in Sections 3.2.1 and 3.2.2.

3.2.1. Multi-scale block

The structure of the multi-scale block is shown in Fig. 2 and includes the following four layers:

The first layer is the convolutional layer which applies a convolution operation to the input with shared weights to extract the features and generate the feature maps. In our proposed model, we apply 3 paralleled convolutional layers as the multi-scale convolution to learn the feature maps of time series, which is aimed at capturing different terms of temporal information. The advantage of the multi-scale convolution is that it produces different sizes of the receptive field to capture more information from the previous layer than the single-scale convolution. The second layer is the concatenate layer which concatenates the feature maps of different convolutional layers on the dimension of channels. The third layer is the batch normalization (BN) layer (Ioffe & Szegedy, 2015) which keeps the same input distribution during the network training by normalizing the feature maps of previous layers. In this way, the gradient becomes larger to avoid the problem of gradient vanishing. Moreover, the larger gradient leads to faster convergence speed, which can greatly accelerate the training process. The last layer is the rectification layer with ReLU (Nair & Hinton, 2010) as the activation function. The advantage of ReLU is that it increases the nonlinear relationship between the layers of the neural network and produces the sparsity to mitigate the occurrence of overfitting problems. More importantly, the combination of BN and ReLU can achieve better performance since it enables more robust learning.

3.2.2. Attention block

The structure of the attention block is depicted in Fig. 2 and includes the following three layers:

The first layer is the global average pooling layer which squeezes the global temporal information into a channel descriptor to generate channel-wise statistics by calculating the mean of all values for each feature map. The input of the attention block is also the output of the multi-scale block X. Let $Z \in \mathbb{R}^C$ be the channel-wise statistics which is generated by shrinking X through the feature map length L_f, where the nth element of Z is calculated by:

$$z_n = \frac{1}{L_f}\sum_{i=1}^{L_f} x_n(i) \tag{1}$$

The last two layers are Fully Connected (FC) layers which achieve the goal of adaptive recalibration to capture the channel-wise weight S by first reducing the feature map channels with a reduction factor r and then rescaling it to the original size. Specifically, the channels of the first FC layer with an activation function of ReLU is $\frac{C}{r}$ and the channels of the second FC layer with an activation function of sigmoid is C, where the nth element of S is obtained by:

$$s_n = \sigma(W_2\delta(W_1 z_n)) \tag{2}$$

where δ and σ refer to the ReLU and sigmoid activation function respectively, $W_1 \in \mathbb{R}^{\frac{C}{r}\times C}$ and $W_2 \in \mathbb{R}^{C\times\frac{C}{r}}$ are parameters of the two FC layers respectively.

The output of the attention block $\tilde{X}$ is obtained by applying the channel-wise weight S to the feature maps, where the nth element of $\tilde{X}$ is calculated by:

$$\tilde{x}_n = s_n \cdot x_n \tag{3}$$

As the whole structure of MACNN is established, we present the experimental results in Section 4.

4. Experiments

4.1. Experiment settings

We choose the same 85 datasets from the UCR archive according to the experiment settings of other approaches (Bagnall et al., 2017; Wang et al., 2017). The length of the datasets ranges from 24 to 2709. The number of classes varies between 2 and 60. The size of the training set varies between 16 and 8926. Moreover, The domain has already been broken down into 7 categories by Bagnall et al. (2017) including device, ECG, image, motion, sensor, simulated, and spectro. Noted that there is no additional data preprocessing since all datasets have already standardized to zero mean and unit variance.

Our framework is implemented based on TensorFlow and run on NVIDIA GTX 1080Ti graphics cards with 3584 cores and 11 GB global memory.[1] The total running time was approximately 30 days because we run the experiments 10 times to average the bias caused by initial weights.

In our proposed CNN structure, the kernel sizes of the multi-scale convolution are fixed to 3, 6, and 12 respectively with the convolution stride fixed to 1, the pooling size fixed to 3 and the stride of the max-pooling layer fixed to 2. An effective optimization method named as Adam (Kingma & Ba, 2015) is adopted to achieve efficient computing. The hyperparameters of MACNN are set to {4, 0.0001, 16, 1500}, which denote the batch size, learning rate, reduction factor and training epoch respectively.

For comparison purposes, we choose 6 competing methods including BOSS, Flat-COTE, Hive-COTE, PROP, FCN, and ResNet. Moreover, to further validate the effectiveness of MACNN, we propose two more models for comparison. One is the Single-scale Attention Convolutional Neural Network (SACNN) which has the same architecture with MACNN except for the convolutional layer in Fig. 2. SACNN consists of a single-scale convolution with the kernel size fixed to 3 and the stride fixed to 1. The other is the Multi-scale Convolutional Neural Network (MCNN) which differs from the architecture of MACNN by removing the attention block. The validation scheme is introduced in Section 4.2, the evaluation metrics of the experiment is introduced in Section 4.3, the experimental results and visualization are displayed in Sections 4.4 and 4.5 respectively.

4.2. Validation scheme

In this section, we carry out 3-fold cross-validation to confirm the final structure and hyperparameters since the training sizes of most datasets are sufficient for deploying. Specifically, the whole validation scheme consists of following two steps:

[1] The final code and detailed results are available on https://github.com/holybaozi/MACNN_Final.

Table 1
Averaged validation error of different hyperparameters combination based on greedy strategy.

Batch size	Learning rate	Reduction factor	Training epoch	Averaged validation error
4	0.001	16	1500	**0.091**
8	0.001	16	1500	0.095
16	0.001	16	1500	0.107
32	0.001	16	1500	0.12
4	0.01	16	1500	0.131
4	0.001	16	1500	0.091
4	0.0001	16	1500	**0.081**
4	0.0001	4	1500	0.097
4	0.0001	8	1500	0.093
4	0.0001	16	1500	**0.081**
4	0.0001	32	1500	0.089
4	0.0001	16	1000	0.092
4	0.0001	16	1500	**0.081**
4	0.0001	16	2000	**0.081**

At the first step, we fix the batch size, learning rate, reduction factor, and training epoch to 16, 0.001, 16, and 1500 respectively as previous researchers (Fawaz et al., 2019; Hu et al., 2019; Wang et al., 2017) used in their study. We choose several datasets with different domains and lengths, including CinCECGTorso, Coffee, CricketX, FacesUCR, ItalyPower, and Mallat mentioned in Section 4.4.1. Then we try different structures which consist of 1 to 6 sections on these datasets (the output section is not included). We apply 3-fold cross-validation on each chosen dataset to obtain the validation error of different structures as shown in Fig. 3. We can find that, as the number of sections increases, the validation error declines first and then rises, shaping like an elbow. This indicates that an underfitting occurs with simple structures and an overfitting occurs with complicated structures. The bottom of the elbow always appears at the point of 3 sections. The structure of 3 sections also achieves the lowest averaged validation error of 0.107. Therefore, we choose the structure of 3 sections as the final structure of our research.

At the second step, we tune the hyperparameters by trying different hyperparameters combinations under the fixed structure of 3 sections. We choose the batch size, learning rate, reduction factor and training epoch from [4, 8, 16, 32], [0.01, 0.001, 0.0001], [4, 8, 16, 32], and [1000, 1500, 2000] respectively. Then, we adopt 3-fold cross-validation to evaluate the performance of each hyperparameters combination by manually changing the value of hyperparameters, and the results are listed in Table 1. We use a greedy strategy based on the previous hyperparameters of {16, 0.001, 16, 1500} to find an appropriate hyperparameters combination. Firstly, we fix the learning rate, reduction factor and training epoch to {0.001, 16, 1500} respectively, then choose different batch size from [4, 8, 16, 32] and apply 3-fold cross-validation to obtain the averaged validation error. We choose 4 as the best batch size since it achieves the lowest averaged validation error. Secondly, we fix the batch size, reduction factor and training epoch to {4, 16, 1500} respectively, then choose different learning rate from [0.01, 0.001, 0.0001] to perform 3-fold cross-validation. The results show that the learning rate of 0.0001 achieves the lowest averaged validation error. Therefore, we choose 0.0001 as the best learning rate. Thirdly, we fix the batch size, learning rate and training epoch to {4, 0.0001, 1500} respectively, then choose different reduction factor from [4, 8, 16, 32] to perform 3-fold cross-validation. We choose 16 as the best reduction factor since it achieves the lowest averaged validation error. Finally, we fix the batch size, learning rate and reduction factor to {4, 0.0001, 16} respectively, then choose different training epoch from [1000, 1500, 2000] to perform 3-fold cross-validation. We find that the training epoch of 1500 and 2000 both achieve the lowest averaged validation error. To reduce training time, we choose 1500 as the best training epoch rather than 2000. Therefore, the final hyperparameters combination is determined as {4, 0.0001, 16, 1500}.

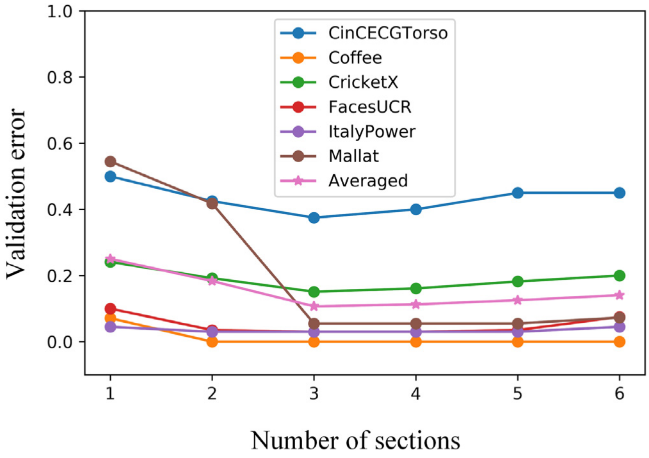


Fig. 3. Validation error of various structures on different datasets.

4.3. Evaluation metrics

We select the following five metrics to evaluate the performance of each method: Wins, Arithmetic Mean Ranking (AMR), Geometric Mean Ranking (GMR), and Mean Error (ME). Noted that, AMR follows the rules of Friedman Test and we apply Friedman Test and Nemenyi Test to compare the performance of all methods. We also take the Binomial Test (BT) and Wilcoxon Signed-rank Test (WST) to measure the significance between MACNN and other methods. We compare the number of parameters of deep learning models to figure out how it affects the classification accuracy. Finally, we record the learning process of each deep learning model for convergence evaluation.

4.4. Results

In this section, a comprehensive evaluation of MACNN and other competing methods is performed.

4.4.1. Evaluation of all methods

As shown in Table 2, MACNN achieves remarkable performance and outperforms the other methods. Specifically, MACNN wins on 44 datasets out of 85 and outperforms others with an average rank of 2.753 and mean error of 0.1271.

Table 2
Summary of error rates of different methods.

Dataset	BOSS	Flat-COTE	Hive-COTE	PROP	FCN	ResNet	SACNN	MCNN	MACNN
Adiac	0.251	0.19	0.185	0.335	**0.143**	0.174	0.187	0.187	0.176
ArrowHead	0.125	0.123	**0.112**	0.14	0.12	0.183	0.166	0.149	0.137
Beef	0.385	0.236	0.277	0.468	0.25	0.233	0.1	0.1	**0.067**
BeetleFly	0.052	0.079	0.041	0.178	0.05	0.2	0.05	**0**	**0**
BirdChicken	0.016	0.059	0.05	0.152	0.05	0.1	**0**	**0**	**0**
Car	0.145	0.101	0.075	0.201	0.083	**0.067**	0.117	0.1	0.083
CBF	0.002	0.002	0.001	0.007	**0**	0.006	0.004	**0**	**0**
ChlorineCon	0.34	0.264	0.275	0.341	0.157	0.172	0.153	0.116	**0.112**
CinCECGTorso	0.1	0.017	**0.012**	0.054	0.187	0.229	0.16	0.149	0.114
Coffee	0.011	**0**	0.002	0.011	**0**	**0**	**0**	**0**	**0**
Computers	0.198	0.23	0.181	0.268	**0.152**	0.176	0.18	0.18	0.168
CricketX	0.236	0.186	0.17	0.199	0.185	0.179	0.254	0.146	**0.138**
CricketY	0.251	0.185	0.163	0.206	0.208	0.195	0.231	0.144	**0.131**
CricketZ	0.224	0.173	0.152	0.196	0.187	0.187	0.215	0.133	**0.121**
DiatomSizeR	0.061	0.075	0.058	0.054	0.07	0.069	**0.023**	**0.023**	**0.023**
DistalPhalanxOutlineAgeGroup	0.186	0.179	0.174	0.232	**0.165**	0.202	0.252	0.245	0.232
DistalPhalanxOutlineCorrect	0.185	0.195	**0.175**	0.232	0.188	0.18	0.185	0.228	0.214
DistalPhalanxTW	0.327	0.307	0.302	0.346	**0.21**	0.26	0.317	0.319	0.302
Earthquakes	0.254	0.253	0.253	0.265	**0.199**	0.214	0.252	0.245	0.245
ECG200	0.11	0.127	0.118	0.119	0.1	0.13	0.1	0.09	**0.08**
ECG5000	0.06	0.054	0.053	0.061	0.059	0.069	0.056	**0.051**	**0.051**
ECGFiveDays	0.017	0.014	0.011	0.153	0.015	0.045	0.075	**0**	**0**
ElectricDevices	0.201	0.117	**0.11**	0.169	0.277	0.272	0.332	0.358	0.304
FaceAll	0.026	0.01	**0.004**	0.024	0.071	0.166	0.054	0.131	0.125
FaceFour	**0.004**	0.15	0.051	0.121	0.068	0.068	0.091	0.034	0.034
FacesUCR	0.049	0.033	**0.016**	0.052	0.052	0.042	0.047	0.021	0.02
FiftyWords	0.298	0.199	0.193	0.179	0.321	0.273	0.253	0.141	**0.13**
Fish	0.031	0.038	0.024	0.087	0.029	0.011	0.017	**0.006**	**0.006**
FordA	0.081	0.045	**0.04**	0.249	0.094	0.072	0.063	0.047	0.045
FordB	0.089	**0.071**	0.073	0.117	0.371	0.1	0.194	0.148	0.126
GunPoint	0.006	0.008	0.003	0.026	**0**	0.007	**0**	**0**	**0**
Ham	0.164	0.195	**0.159**	0.237	0.238	0.219	0.286	0.171	0.171
HandOutlines	0.097	0.106	0.088	0.12	0.224	0.139	0.076	0.051	**0.049**
Haptics	0.541	0.483	0.47	0.549	**0.449**	0.494	0.545	0.458	0.458
Herring	0.395	0.368	0.366	0.434	**0.297**	0.406	0.328	0.328	0.312
InlineSkate	0.497	**0.474**	**0.474**	0.524	0.589	0.635	0.533	0.504	0.504
InsectWingbeatSound	0.49	0.361	0.365	0.419	0.598	0.469	0.547	0.368	**0.353**
ItalyPower	0.134	0.03	0.032	0.049	0.03	0.04	**0.028**	**0.028**	**0.028**
LargeKitchenAppliances	0.163	0.1	**0.077**	0.184	0.104	0.107	0.093	0.08	0.08
Lightning2	0.19	0.215	0.203	**0.165**	0.197	0.246	0.213	0.197	0.18
Lightning7	0.334	0.201	0.189	0.237	**0.137**	0.164	0.192	0.151	**0.137**
MALLAT	0.051	0.026	0.025	0.039	**0.02**	0.021	0.025	0.024	**0.02**
Meat	0.02	0.019	0.013	0.022	0.033	**0**	0.017	0.017	**0**
MedicalImages	0.285	0.215	**0.185**	0.24	0.208	0.228	0.234	0.238	0.217
MiddlePhalanxOutlineAgeGroup	0.334	0.278	0.295	0.391	**0.232**	0.24	0.416	0.409	0.383
MiddlePhalanxOutlineCorrect	0.192	0.199	0.191	0.218	0.205	0.207	0.168	0.179	**0.162**
MiddlePhalanxTW	0.463	0.413	0.429	0.475	**0.388**	0.393	0.409	0.468	0.409
MoteStrain	0.154	0.098	0.053	0.125	**0.05**	0.105	0.082	0.082	0.082
NonInvThorax1	0.159	0.071	0.068	0.151	**0.039**	0.052	0.053	0.051	0.051
NonInvThorax2	0.096	0.054	0.048	0.086	**0.045**	0.049	0.054	**0.045**	**0.045**
OliveOil	0.13	**0.099**	0.102	0.121	0.167	0.133	0.167	0.1	0.1
OSULeaf	0.033	0.051	0.03	0.188	**0.012**	0.021	0.029	0.029	0.021
PhalangesOutlinesCorrect	0.179	0.217	0.179	0.22	0.174	0.175	0.176	0.17	**0.167**
Phoneme	0.744	0.638	**0.615**	0.701	0.655	0.676	0.664	0.678	0.66
Plane	0.002	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
ProximalPhalanxOutlineAgeGroup	0.181	0.152	0.152	0.195	0.151	0.151	0.151	0.151	**0.146**
ProximalPhalanxOutlineCorrect	0.133	0.129	0.124	0.161	0.1	0.082	0.079	0.079	**0.076**
ProximalPhalanxTW	0.227	**0.185**	0.188	0.241	0.19	0.193	0.224	0.224	0.208
RefrigerationDevices	0.215	0.258	**0.199**	0.324	0.467	0.472	0.429	0.424	0.411
ScreenType	0.414	0.349	**0.289**	0.446	0.333	0.293	0.365	0.371	0.352
ShapeletSim	**0**	0.036	0.009	0.173	0.133	**0**	0.011	**0**	**0**
ShapesAll	0.091	0.089	0.074	0.114	0.102	0.088	0.1	0.065	**0.06**
SmallKitchenAppliances	0.25	0.212	**0.163**	0.297	0.197	0.203	0.219	0.205	0.192
SonyAIBORobot	0.103	0.101	0.113	0.206	0.032	**0.015**	0.017	0.017	**0.015**
SonyAIBORobotII	0.112	0.04	0.055	0.13	0.038	0.038	**0.024**	0.04	0.038
StarLightCurves	0.022	0.02	**0.019**	0.059	0.033	0.029	0.026	0.024	0.022
Strawberry	0.03	0.037	0.03	0.041	0.031	0.042	0.027	**0.024**	**0.024**
SwedishLeaf	0.082	0.033	**0.031**	0.084	0.034	0.042	0.037	0.037	0.037
Symbols	0.039	0.047	0.034	0.043	0.038	0.128	0.021	0.021	**0.02**
SyntheticControl	0.032	0.001	**0**	0.006	0.01	**0**	**0**	**0**	**0**
ToeSegmentation1	0.071	0.066	0.045	0.212	0.031	0.035	**0.026**	**0.026**	**0.026**
ToeSegmentation2	0.04	0.049	**0.034**	0.093	0.085	0.138	0.069	0.054	0.054
Trace	**0**	**0**	**0**	0.004	**0**	**0**	**0**	**0**	**0**
TwoLeadECG	0.016	0.018	0.007	0.042	**0**	**0**	0.001	**0**	**0**
TwoPatterns	0.009	**0**	**0**	**0**	0.103	**0**	0.003	**0**	**0**

(*continued on next page*)

Table 2 (*continued*).

Dataset	BOSS	Flat-COTE	Hive-COTE	PROP	FCN	ResNet	SACNN	MCNN	MACNN
UWaveAll	0.056	0.035	**0.03**	0.032	0.174	0.132	0.128	0.054	0.04
UWaveX	0.247	0.17	0.162	0.195	0.246	0.213	0.204	0.169	**0.16**
UWaveY	0.339	0.234	0.225	0.27	0.275	0.332	0.3	0.229	**0.219**
UWaveZ	0.305	0.241	0.222	0.274	0.271	0.245	0.246	0.223	**0.212**
Wafer	0.001	0.001	**0**	0.003	0.003	0.003	**0**	**0**	**0**
Wine	**0.088**	0.097	**0.088**	0.113	0.111	0.204	0.167	0.148	0.13
WordSynonyms	0.341	0.252	0.252	**0.222**	0.42	0.368	0.394	0.255	0.234
Worms	0.265	0.275	0.266	0.356	0.331	0.381	**0.13**	0.156	0.143
WormsTwoClass	0.19	0.215	0.216	0.283	0.271	0.265	0.156	0.182	**0.13**
Yoga	0.09	0.102	0.083	0.115	0.155	0.142	0.104	0.082	**0.08**
Wins	4	8	25	4	21	10	12	20	**44**
AMR	6.376	5.047	3.624	7.418	5	5.582	5.341	3.859	**2.753**
GMR	5.864	4.529	2.977	6.914	4.061	4.964	4.822	3.42	**2.293**
ME	0.1666	0.1421	0.1308	0.1882	0.156	0.1615	0.1544	0.136	**0.1271**

Table 3
Pairwise comparison of all models against MACNN.

Model	better	tie	worse	p(BT)	p(WST)
BOSS	14	3	68	1.13E−09	4.24E−08
Flat-COTE	20	5	60	8.58E−06	7.74E−05
Hive-COTE	26	6	53	3.18E−03	7.40E−02
PROP	8	3	74	1.65E−14	3.70E−11
FCN	23	11	51	1.52E−03	1.02E−04
ResNet	12	11	62	2.86E−09	7.48E−08
SACNN	4	13	68	4.62E−16	7.39E−11
MCNN	0	32	53	2.22E−16	2.37E−10

Table 4
Critical values for Nemenyi test.

#classifiers	2	3	4	5	6	7	8	9	10
$q_{0.05}$	1.96	2.343	2.569	2.728	2.85	2.949	3.031	3.102	3.164
$q_{0.1}$	1.645	2.052	2.291	2.459	2.589	2.693	2.78	2.855	2.92

For intuitive comparison, we separate all methods into several clusters according to their min–max normalization of ME. As shown in Fig. 4, the best group includes MACNN, Hive-COTE, MCNN, and Flat-COTE, which achieves remarkable performance than the other groups. The second group includes SACNN, FCN, ResNet, and BOSS. The last group only includes PROP. More importantly, in the best group, MACNN achieves the lowest ME and outperforms other traditional and deep learning based methods.

For more comparison, we count the number of datasets which performs better, tie, or worse than MACNN and apply the Binomial Test (BT) and Wilcoxon Signed-rank Test (WST) to measure the significance of difference between MACNN and other approaches. Corresponding p-values are listed in Table 3, which provides evidence that MACNN is the most accurate classifier and achieves much better performance than other methods.

To further evaluate the performance of all methods, we perform Friedman Test to compare multiple classifiers as advised by the research of Demšar (2006). The Friedman statistic (Iman & Davenport, 1980) F_F is calculated by:

$$F_F = \frac{(N-1)\,\chi_F^2}{N\,(k-1) - \chi_F^2} \tag{4}$$

$$\chi_F^2 = \frac{12N}{k\,(k+1)}\left[\sum_{j=1}^{k} R_j^2 - \frac{k\,(k+1)^2}{4}\right] \tag{5}$$

where k and N refer to the number of methods and datasets respectively, R_j refers to the value of AMR of each method. With 9 methods and 85 data sets, F_F is distributed according to the F distribution with $9-1=8$ and $(9-1)\times(85-1)=672$ degrees of freedom. The critical value of $F(8,672)$ for $\alpha=0.05$ is 1.952. However, the value of F_F calculated by the equations above is 31.707. Therefore, we reject the null-hypothesis and proceed with Nemenyi Test later. The critical difference (CD) of Nemenyi Test is calculated by:

$$CD = q_\alpha\sqrt{\frac{k\,(k+1)}{6N}} \tag{6}$$

where q_α is the critical value which can be looked up in Table 4. With 9 algorithms and 85 data sets, q_α for $\alpha = 0.05$ is 3.102 and corresponding CD is 1.303. Therefore, we can conclude that MACNN achieves better performance than Hive-COTE ($3.624 - 2.753 = 0.871 < 1.303$) and MCNN($3.859 - 2.753 = 1.106 < 1.303$), and achieves significantly better performance than other methods (for example: Flat-COTE ($5.047 - 2.753 = 2.294 > 1.303$)).

For intuitive comparison, we show the scatter plots of pairwise comparison of all models against MACNN in Fig. 5. Each dot represents the error rate of the two classifiers on one dataset, and the farther a dot is located from the diagonal line, the greater the difference of accuracy. Moreover, a dot above the diagonal line indicates that MACNN is more accurate than the rivaling method, and vise versa. Noted that, the "best classifier" indicates the best results from BOSS, Flat-COTE, Hive-COTE, PROP, FCN, and ResNet. Fig. 5 shows that MCNN is better than other methods since most of the dots are above the diagonal line. More importantly, there is a huge gap between MACNN and the rivaling method on a lot of datasets except for MCNN, which proves that the MAM and the structure of MACNN are effective to solve TSC tasks, and also indicates that the attention mechanism is a bit helpful for improving the accuracy of classification.

To evaluate the performance of each method on different characteristics of the datasets, we divide the datasets into several groups according to their domain and length, which is first defined by Bagnall et al. (2018), and then use AMR to evaluate the performance of different approaches in each group. Noted that, the numerical value in bracket represents the probability of best performance with each method in different groups, which is calculated by the number of Wins divided by the number of datasets in each group.

The performance of different approaches grouped by domain is listed in Table 5. We can find that MACNN achieves the best performance across different domains except for device domain. The datasets of device group are almost long time series and all methods achieve higher error rate on them compare to other groups, which means the TSC task in device domain is more complicated. However, Hive-COTE is suitable to deal with such problems since its hierarchical structure with probabilistic voting by multiple classifiers. Another interesting discovery is that MACNN achieves the best performance on simulated group with the probability of best performance reaches 100%. The datasets of simulated group are synthesized by adding shift, trend, pulse,

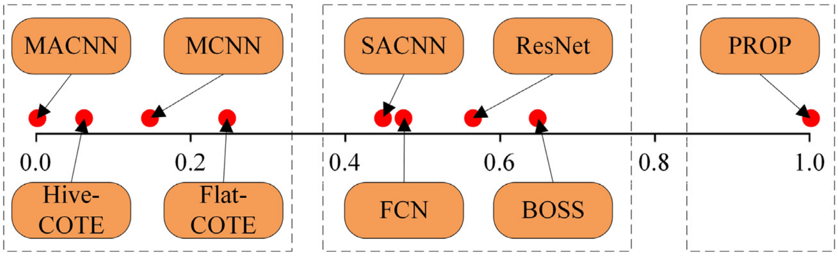


Fig. 4. Models grouping by the normalized ME.

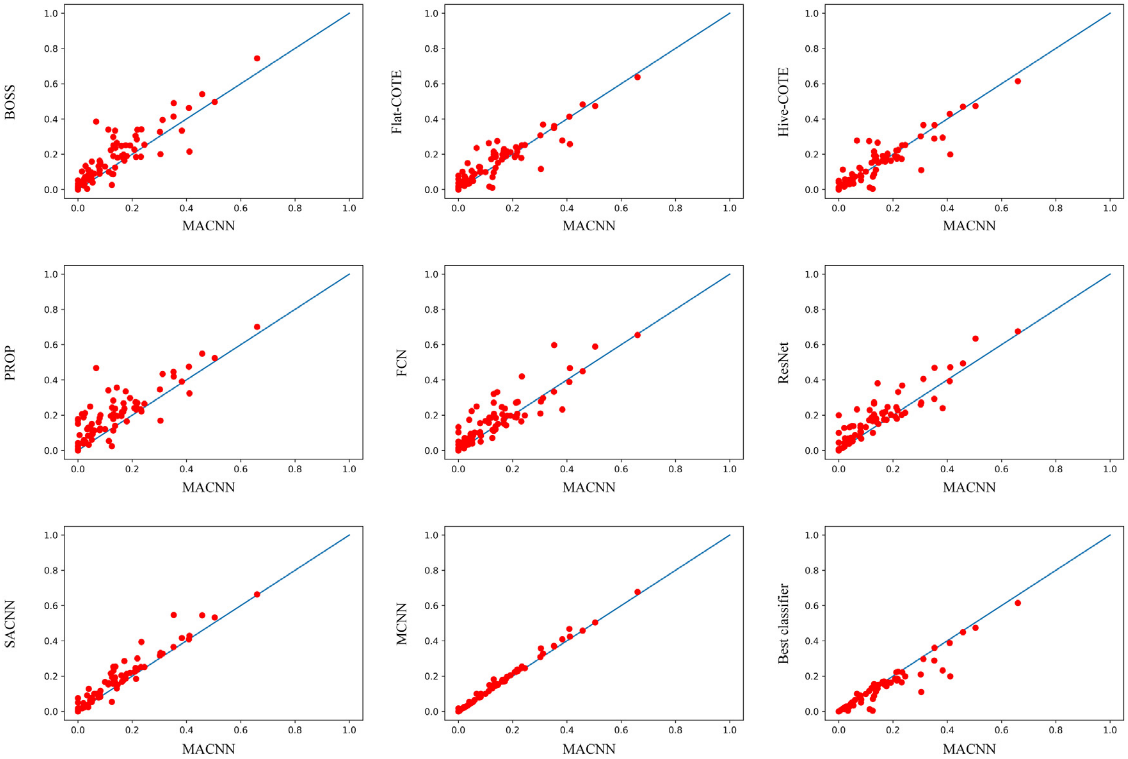


Fig. 5. Scatter plots of pairwise comparison of all models against MACNN.

Table 5
The performance of different approaches grouped by domain.

Domain	BOSS	Flat-COTE	Hive-COTE	PROP	FCN	ResNet	SACNN	MCNN	MACNN
Device	6.167(0)	4.667(0)	**1.383(0.833)**	7.167(0)	4.5(0.167)	5(0)	6.083(0)	5.667(0)	3.917(0)
ECG	6.714(0)	5.643(0)	4.143(0.143)	7.429(0)	4(0.429)	6.5(0.143)	5.714(0)	2.571(0.571)	**2.286(0.714)**
Image	6.052(0.034)	5.448(0.034)	3.638(0.207)	7.552(0.034)	4.603(0.241)	5.517(0)	5.069(0.069)	4.224(0.138)	**2.862(0.448)**
Motion	6.643(0)	4.607(0.071)	3.179(0.214)	7(0)	6.071(0.143)	6.75(0)	5.679(0.214)	3(0.143)	**2.071(0.643)**
Sensor	6.7(0.067)	4.767(0.2)	4.067(0.333)	7.633(0.133)	4.733(0.333)	4.9(0.267)	5(0.333)	4.167(0.267)	**3.033(0.467)**
Simulated	7(0.167)	5.833(0.167)	4.667(0.333)	7.583(0.167)	5.417(0.333)	4.167(0.5)	5.25(0.167)	2.833(0.667)	**2.25(1)**
Spectro	5.357(0.143)	4.214(0.286)	4(0.286)	7.214(0)	6.286(0.143)	5.571(0.286)	5.429(0.143)	3.429(0.286)	**2.643(0.571)**

white noise, and other ways, which is less complicated than the real-world datasets. Moreover, the combination of multi-scale convolution and attention mechanism helps to improve the recognition ability of the network. Therefore, MACNN shows an overwhelming advantage on simulated group.

To investigate how the length of datasets affects the performance of different approaches, we group the performance by length which is shown in Table 6. We find that MACNN performs better than others in different ranges of length, which indicates that multi-scale convolution is fit for varies lengths of datasets since it aims to capture the short-term, mid-term, and long-term dependencies of time series. Noted that, MACNN outperforms other approaches by a large margin when the length of datasets between 300 and 700. This can be explained by the receptive field of different models. If the length is too short, the receptive field of other CNN models may nearly cover most areas of the time series, which captures roughly the same features as MACNN. However, if the length is too long, both MACNN and other models can only capture a small sequence, which leads to limited improvement on longer datasets. Therefore, the outstanding performance of MACNN indicates that the length of datasets between 300 and 700 is more appropriate for MACNN.

In summary, the above results show that MACNN is an outstanding approach that achieves the best performance in a vast majority of groups. In traditional methods, Hive-COTE achieves better performance than BOSS, Flat-COTE, and PROP, while the

Table 6
The performance of different approaches grouped by length.

Length	BOSS	Flat-COTE	Hive-COTE	PROP	FCN	ResNet	SACNN	MCNN	MACNN
< 300	6.588(0.05)	4.825(0.125)	3.9(0.325)	7.438(0.075)	4.388(0.3)	5.338(0.175)	4.95(0.225)	4.275(0.35)	**3.125(0.575)**
300–700	6.161(0.071)	5.571(0.071)	3.643(0.107)	7.429(0.036)	5.786(0.143)	5.536(0.107)	5.786(0.0.071)	3.018(0.179)	**2.071(0.607)**
> 700	6.206(0)	4.647(0.059)	**2.941(0.529)**	7.353(0)	5.088(0.294)	6.176(0)	5.441(0.059)	4.206(0.059)	**2.941(0.235)**

performance of FCN and ResNet is better than boss and PROP but worse than Flat-COTE and Hive-COTE, which demonstrates the feasibility of deep learning based methods for the TSC task. In addition, MACNN outperforms other deep leaning based methods by a large margin and shows better performance than Hive-COTE, which suggests that the deep learning based methods can catch up with traditional methods. More importantly, the remarkable performance of MACNN proves the effectiveness of the combination of multi-scale convolution and attention mechanism, and shows that deep learning based approach is a breakthrough to improve the accuracy of TSC task.

4.4.2. Evaluation of deep learning based models

Since the initial weights of deep learning based models may result in the bias, Fawaz et al. (2019) average the accuracy over 10 runs with random initialization of weights, which helps to assess the importance and impact of the model. Therefore, we take the same strategy to run the experiments 10 times and compare the averaged error rate with the top two deep learning models according to the study of Fawaz et al. The results are shown in Table 7. Noted that, the values in the parentheses refer to the standard deviation and we add one more evaluation metric named as Mean Standard Deviation (MSD) to measure the stability of the model. Our empirical study strongly suggests using MACNN since it achieves the lowest error rate on 63 datasets out of 85. Moreover, MACNN achieves the lowest MSD, which shows better stability than other models.

For more comparison, we count the number of datasets that performs better, significantly better, worse or significantly worse than MACNN, and the results are listed in Table 8. Noted that if we claim model A is significantly better or worse than model B on a dataset, this means that the average error rate of model A is 0.05 lower or higher than model B. As shown in Table 8, MACNN achieves remarkable performance than FCN and ResNet, which proves the effectiveness of MACNN. Moreover, comparing SACNN and MCNN with MACNN, SACNN achieves worse performance than MACNN on 75 datasets out of 85, and significantly worse on 17 datasets, which shows the importance of the multi-scale convolution; MCNN achieves worse performance than MACNN on 68 datasets with 3 significantly worse, which demonstrate that the attention mechanism can slightly improve the classification accuracy.

To figure out how MACNN improves the performance, we calculate the number of parameters of deep learning models. Since the last layer of these models is the global average pooling layer which gives rather a small contribution to the parameters, we only calculate the parameters of CNN layers. The number of parameters of the ith CNN layer is p, which is calculated as follows:

$$p_i = k_i \times c_{i-1} \times c_i + c_i \tag{7}$$

where k_i is the kernel size of ith CNN layer, c_{i-1} and c_i are the number of channels of $i - 1^{th}$ and ith CNN layer respectively. Moreover, the number of parameters of the attention block in ith CNN layer is a, which is calculated as follows:

$$a_i = 2 \times c_i \times \frac{c_i}{r} \tag{8}$$

where r is the reduction factor of the attention block.

The number of parameters of different deep learning models is shown in Table 9. We find that FCN, ResNet, and SACNN have the same order of magnitude of parameters, which may explain their higher average error rates caused by insufficient model complexity. On the contrary, MACNN and MCNN achieve lower average error rates with higher model complexity. Therefore, comparing MACNN with other models, the remarkable performance of MACNN can be explained by two reasons. One is the higher model complexity offered by a larger number of parameters. Since we propose the multi-scale block and attention block which increases the number of parameters, it becomes easier to result in overfitting and worse generalization ability on a few datasets. However, comparing to the previous research which usually applies simple structures such as FCN and ResNet, we achieve much better performance on most datasets due to the higher model complexity. The other is the model generalization ability increased by pooling layers. Unlike FCN and ResNet which keep the length of time series unchanged throughout the whole process of convolution, we utilize the max-pooling layer to reduce the redundant features created by a large overlap area of convolution. The advantage of the max-pooling layer is that it reduces time consumption and increases model generalization ability by reducing the number of parameters while keeping the main characteristics. Although it may result in worse performance on certain datasets due to the local deformation, in most cases, small local deformation has little effect on the classification results.

For convergence evaluation, we select two datasets on which there are obvious differences between different models, then record the error rate of each epoch to observe the convergence process. As shown in Fig. 6, we can divide all models into 3 groups. The first group includes MACNN and MCNN which converges to lower error rate with faster convergence rate compared to other groups. The second group consists of SACNN and ResNet which achieves medium performance. The last group is FCN which achieves the highest error rate with the slowest convergence rate. Therefore, the convergence evaluation is consistent with the previous analysis of model complexity.

The binary classification is a fundamental problem in TSC task, and in our experiments, 31 out of 85 are binary classification tasks. Therefore, to intuitively observe the gap between the performance of different deep learning models on binary classification, we draw ROC curves of each model on the dataset of Ham and use AUC-ROC to evaluate the performance. As shown in Fig. 7, MACNN achieves the largest AUC-ROC value and outperforms other models by a large margin, which suggests that MACNN is the best deep learning model on binary classification.

In conclusion, the experimental results prove that our proposed model achieves significant progress compared to state-of-the-art traditional and deep learning based TSC methods. To further validate the effectiveness of MACNN, more details are given in Section 4.5.

4.5. Visualization

We investigate the use of Class Activation Map (CAM) (Zhou et al., 2016) which is proposed to find the most contributing area of an image. Later, Wang et al. (2017) and Fawaz et al. (2019) introduced a one-dimensional CAM for visualization to find out

Table 7
Summary of averaged error rates and standard deviation of deep learning models.

Dataset	FCN	ResNet	SACNN	MCNN	MACNN
Adiac	**0.156(0.007)**	0.171(0.006)	0.202(0.011)	0.197(0.007)	0.188(0.011)
ArrowHead	0.157(0.015)	0.155(0.012)	0.186(0.024)	0.154(0.008)	**0.145(0.007)**
Beef	0.303(0.04)	0.247(0.042)	0.137(0.046)	0.117(0.022)	**0.093(0.025)**
BeetleFly	0.14(0.097)	0.15(0.024)	0.075(0.046)	0.01(0.02)	**0.005(0.015)**
BirdChicken	0.045(0.037)	0.115(0.053)	0.03(0.04)	0.02(0.033)	**0.01(0.03)**
Car	0.095(0.014)	**0.075(0.014)**	0.139(0.024)	0.108(0.009)	0.09(0.008)
CBF	0.006(0.001)	0.005(0.003)	0.01(0.006)	**0.001(0.001)**	**0.001(0.001)**
ChlorineCon	0.186(0.009)	0.156(0.01)	0.161(0.005)	0.12(0.003)	**0.118(0.004)**
CinCECGTorso	0.176(0.012)	0.174(0.024)	0.172(0.012)	0.159(0.013)	**0.123(0.011)**
Coffee	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
Computers	**0.178(0.01)**	0.185(0.012)	0.195(0.019)	0.193(0.013)	0.186(0.017)
CricketX	0.208(0.007)	0.209(0.006)	0.265(0.012)	0.159(0.014)	**0.146(0.008)**
CricketY	0.213(0.012)	0.197(0.008)	0.25(0.024)	0.155(0.013)	**0.137(0.007)**
CricketZ	0.189(0.01)	0.188(0.014)	0.225(0.013)	0.143(0.012)	**0.13(0.011)**
DiatomSizeR	0.687(0.036)	0.699(0.002)	0.039(0.022)	0.032(0.011)	**0.03(0.011)**
DistalPhalanxOutlineAgeGroup	0.29(0.013)	0.283(0.013)	0.269(0.023)	0.252(0.011)	**0.238(0.013)**
DistalPhalanxOutlineCorrect	0.24(0.015)	0.229(0.01)	**0.208(0.028)**	0.235(0.009)	0.225(0.012)
DistalPhalanxTW	**0.31(0.021)**	0.335(0.016)	0.327(0.012)	0.321(0.015)	0.318(0.011)
Earthquakes	0.273(0.017)	0.288(0.02)	0.263(0.012)	0.253(0.011)	**0.252(0.007)**
ECG200	0.111(0.01)	0.126(0.019)	0.102(0.004)	0.095(0.005)	**0.085(0.005)**
ECG5000	0.06(0.001)	0.066(0.002)	0.061(0.004)	0.055(0.004)	**0.054(0.003)**
ECGFiveDays	0.013(0.003)	0.025(0.019)	0.087(0.019)	0.006(0.007)	**0.003(0.006)**
ElectricDevices	0.298(0.012)	**0.271(0.009)**	0.354(0.016)	0.363(0.01)	0.309(0.009)
FaceAll	**0.055(0.009)**	0.161(0.02)	0.067(0.017)	0.142(0.013)	0.131(0.009)
FaceFour	0.072(0.009)	0.045(0)	0.113(0.026)	**0.044(0.01)**	**0.044(0.009)**
FacesUCR	0.054(0.002)	0.045(0.004)	0.051(0.003)	0.026(0.005)	**0.024(0.005)**
FiftyWords	0.373(0.061)	0.26(0.015)	0.265(0.014)	0.153(0.01)	**0.138(0.01)**
Fish	0.042(0.006)	0.021(0.008)	0.023(0.006)	0.012(0.01)	**0.011(0.011)**
FordA	0.096(0.002)	0.08(0.004)	0.068(0.009)	0.052(0.007)	**0.05(0.008**)
FordB	0.122(0.006)	**0.087(0.003)**	0.206(0.013)	0.15(0.004)	0.132(0.005)
GunPoint	**0(0)**	0.009(0.007)	0.003(0.003)	0.002(0.003)	0.002(0.003)
Ham	0.282(0.014)	0.243(0.027)	0.292(0.014)	**0.178(0.007)**	**0.178(0.007)**
HandOutlines	0.194(0.079)	0.089(0.014)	0.086(0.008)	0.068(0.014)	**0.062(0.014)**
Haptics	0.52(0.024)	0.481(0.012)	0.565(0.021)	**0.468(0.011)**	**0.468(0.011)**
Herring	0.392(0.077)	0.381(0.038)	0.366(0.036)	0.344(0.017)	**0.325(0.014)**
InlineSkate	0.661(0.008)	0.627(0.009)	0.554(0.022)	**0.517(0.016)**	**0.517(0.012)**
InsectWingbeatSound	0.607(0.006)	0.493(0.009)	0.561(0.012)	0.373(0.007)	**0.357(0.005)**
ItalyPower	0.039(0.003)	0.037(0.004)	0.033(0.004)	0.033(0.003)	**0.032(0.004)**
LargeKitchenAppliances	0.098(0.004)	0.1(0.005)	0.098(0.004)	0.087(0.007)	**0.085(0.004)**
Lightning2	0.261(0.014)	0.23(0.017)	0.241(0.038)	0.218(0.031)	**0.197(0.018)**
Lightning7	0.173(0.023)	**0.155(0.02)**	0.222(0.032)	0.177(0.021)	0.159(0.016)
Mallat	0.033(0.009)	0.028(0.003)	0.031(0.007)	0.029(0.008)	**0.025(0.005)**
Meat	0.147(0.069)	0.032(0.025)	0.047(0.033)	0.037(0.028)	**0.022(0.026)**
MedicalImages	**0.221(0.004)**	0.23(0.007)	0.248(0.017)	0.244(0.016)	0.226(0.012)
MiddlePhalanxOutlineAgeGroup	0.447(0.018)	0.431(0.021)	0.433(0.018)	0.419(0.021)	**0.391(0.012)**
MiddlePhalanxOutlineCorrect	0.199(0.01)	0.191(0.012)	0.181(0.011)	0.187(0.008)	**0.171(0.01)**
MiddlePhalanxTW	0.488(0.018)	0.516(0.02)	0.432(0.029)	0.482(0.016)	**0.419(0.005)**
MoteStrain	**0.063(0.005)**	0.072(0.005)	0.092(0.009)	0.091(0.007)	0.09(0.007)
NonInvThorax1	**0.044(0.003)**	0.055(0.003)	0.057(0.004)	0.055(0.003)	0.055(0.003)
NonInvThorax2	**0.047(0.003)**	0.054(0.003)	0.058(0.005)	0.049(0.003)	**0.047(0.003)**
OliveOil	0.277(0.166)	0.17(0.085)	0.217(0.128)	0.153(0.069)	**0.15(0.065)**
OSULeaf	0.023(0.009)	**0.021(0.008)**	0.038(0.009)	0.03(0.008)	0.029(0.008)
PhalangesOutlinesCorrect	0.18(0.005)	**0.161(0.012)**	0.185(0.009)	0.176(0.008)	0.174(0.005)
Phoneme	0.675(0.005)	0.666(0.007)	0.673(0.01)	0.689(0.011)	**0.664(0.004)**
Plane	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
ProximalPhalanxOutlineAgeGroup	0.169(0.013)	**0.147(0.008)**	0.162(0.008)	0.16(0.006)	0.153(0.006)
ProximalPhalanxOutlineCorrect	0.097(0.007)	**0.079(0.006)**	0.095(0.008)	0.086(0.007)	0.084(0.006)
ProximalPhalanxTW	0.233(0.009)	0.22(0.017)	0.237(0.014)	0.235(0.014)	**0.213(0.011)**
RefrigerationDevices	0.492(0.01)	0.475(0.025)	0.448(0.022)	0.437(0.02)	**0.421(0.015)**
ScreenType	0.375(0.016)	0.378(0.014)	0.389(0.018)	0.4(0.018)	**0.365(0.016)**
ShapeletSim	0.276(0.056)	0.221(0.15)	0.024(0.015)	**0.006(0.008)**	**0.006(0.009)**
ShapesAll	0.105(0.004)	0.079(0.004)	0.103(0.005)	0.074(0.017)	**0.067(0.011)**
SmallKitchenAppliances	0.217(0.013)	0.214(0.008)	0.227(0.01)	0.216(0.016)	**0.204(0.011)**
SonyAIBORobot	0.04(0.007)	0.042(0.013)	0.027(0.008)	0.024(0.01)	**0.021(0.007)**
SonyAIBORobotII	**0.021(0.005)**	0.022(0.005)	0.026(0.002)	0.044(0.004)	0.041(0.003)
StarLightCurves	0.039(0.009)	0.028(0.003)	0.031(0.006)	0.027(0.002)	**0.024(0.002)**
Strawberry	0.028(0.003)	**0.019(0.004)**	0.03(0.003)	0.028(0.006)	0.027(0.004)
SwedishLeaf	**0.031(0.005)**	0.044(0.004)	0.041(0.004)	0.039(0.002)	0.039(0.001)
Symbols	0.045(0.01)	0.094(0.023)	0.035(0.021)	0.027(0.009)	**0.025(0.009)**
SyntheticControl	0.015(0.003)	0.002(0.002)	**0.001(0.001)**	**0.001(0.001)**	**0.001(0.001)**
ToeSegmentation1	0.039(0.005)	0.037(0.006)	0.042(0.006)	0.036(0.008)	**0.034(0.006)**
ToeSegmentation2	0.12(0.033)	0.094(0.017)	0.096(0.024)	**0.061(0.006)**	**0.061(0.005)**
Trace	**0(0)**	**0(0)**	**0(0)**	**0(0)**	**0(0)**
TwoLeadECG	**0(0)**	**0(0)**	0.001(0.001)	0.001(0.001)	0.001(0.001)
TwoPatterns	0.129(0.003)	**0(0)**	0.013(0.007)	**0(0)**	**0(0)**

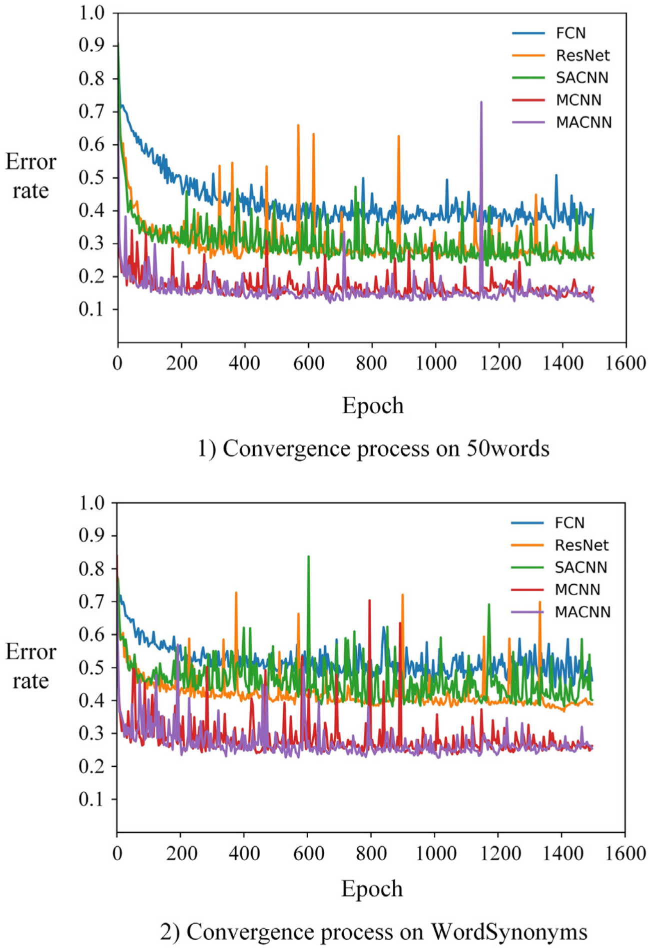
(*continued on next page*)

Table 7 (*continued*).

Dataset	FCN	ResNet	SACNN	MCNN	MACNN
UWaveAll	0.183(0.003)	0.14(0.004)	0.134(0.012)	0.059(0.012)	**0.044(0.004)**
UWaveX	0.246(0.004)	0.22(0.004)	0.214(0.012)	0.179(0.011)	**0.165(0.006)**
UWaveY	0.361(0.006)	0.33(0.007)	0.307(0.005)	0.234(0.004)	**0.221(0.003)**
UWaveZ	0.274(0.005)	0.25(0.004)	0.255(0.006)	0.231(0.007)	**0.216(0.004)**
Wafer	0.003(0)	**0.001(0.001)**	**0.001(0.001)**	**0.001(0.001)**	**0.001(0.001)**
Wine	0.413(0.083)	0.256(0.085)	0.228(0.098)	0.194(0.061)	**0.161(0.058)**
WordSynonyms	0.436(0.012)	0.378(0.015)	0.407(0.012)	0.268(0.011)	**0.242(0.009)**
Worms	0.235(0.022)	0.209(0.025)	**0.142(0.018)**	0.172(0.018)	0.151(0.012)
WormsTwoClass	0.274(0.027)	0.253(0.033)	0.164(0.019)	0.196(0.019)	**0.141(0.014)**
Yoga	0.161(0.007)	0.13(0.009)	0.116(0.007)	0.086(0.005)	**0.084(0.004)**
Wins	15	15	7	13	**63**
AMR	3.869	3.256	3.732	2.589	**1.554**
GMR	3.448	2.939	3.513	2.425	**1.391**
ME	0.1915	0.1751	0.1678	0.1454	**0.135**
MSD	0.0171	0.0148	0.0161	0.0113	**0.0094**

Table 8
Pairwise comparison of deep learning models against MACNN.

Model	Better	Significantly better	Worse	Significantly worse
FCN	14	1	67	34
ResNet	14	0	65	24
SACNN	4	1	75	17
MCNN	0	0	68	3



1) Convergence process on 50words

2) Convergence process on WordSynonyms

Fig. 6. Convergence process of different deep learning models.

the subsequences of time series which contribute the most for classification. Let $f_m(t)$ represent the activation of the filter $m \in [1, M]$ in the last convolutional layer at time stamp $t \in [1, T]$. Let ω_m^c be the weight between the mth filter and the output neuron of class c. Since the global average pooling is used, then the class score S_c is calculated as follows:

Table 9
The number of parameters of deep learning models.

Layer	FCN	ResNet	SACNN	MCNN	MACNN
Cov1	1152	576	768	1536	6144
Cov2	164 096	20 544	12 864	86 208	90 816
Cov3	98 432	12 352	24 704	172 416	190 848
Cov4		65 664	51 328	344 448	362 880
Cov5		82 048	106 752	688 896	762 624
Cov6		49 280	205 056	1 377 024	1 450 752
Cov7		131 200			
Cov8		82 048			
Cov9		49 280			
Total	263 680	492 992	401 472	2 670 528	2 864 064

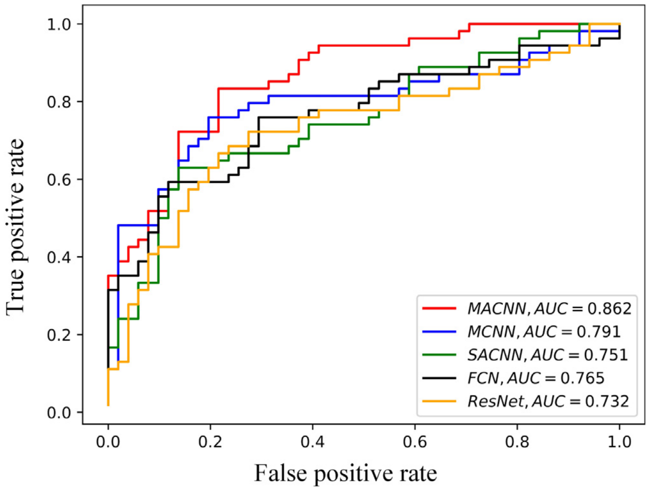


Fig. 7. ROC curves of different deep learning models.

$$S_c = \sum_m \omega_m^c \sum_t f_m(t) \tag{9}$$

The input S_c can also be rewritten as follows:

$$S_c = \sum_t \sum_m \omega_m^c f_m(t) \tag{10}$$

Finally, we define CAM_c as CAM for class c, where each time stamp is obtained by:

$$CAM_c(t) = \sum_m \omega_m^c f_m(t) \tag{11}$$

Therefore, $CAM_c(t)$ directly shows the importance of activation at time stamp t which generates the classification result c. By

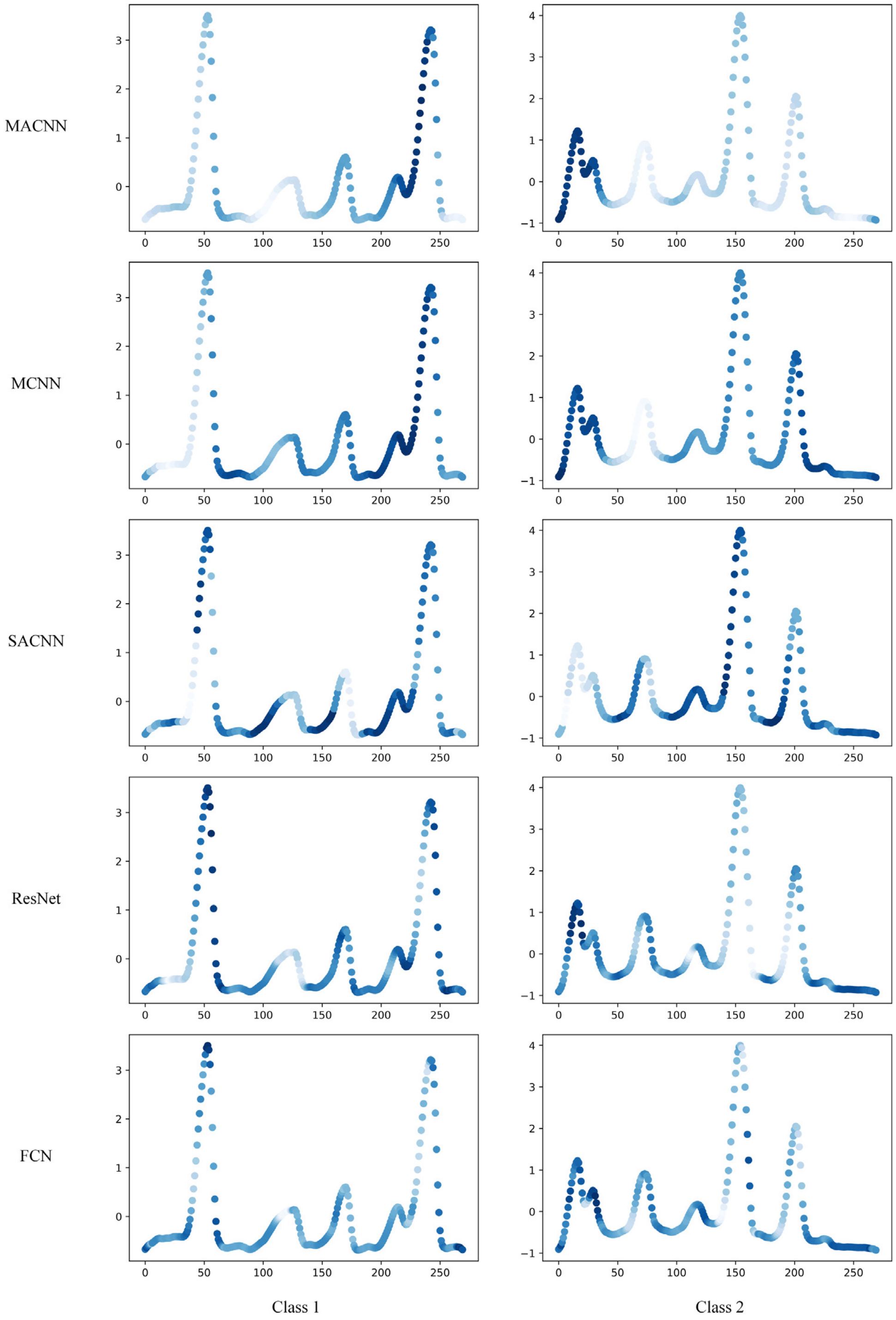


Fig. 8. CAM results of different models on the WordSynonyms dataset.

simply upsampling CAM to the size of raw time series, the most important subsequences can be identified.

We apply CAM on the WordSynonyms and GunPoint datasets to figure out the effect of the multi-scale convolution and attention mechanism. There are several reasons that we choose the two datasets for visualization. For the WordSynonyms dataset, since MACNN outperforms other models by a large margin, we intend to find out more interpretable explanations by comparing the CAM results of MACNN with others. For the GunPoint dataset, since the discriminative subsequence of the GunPoint dataset is found and interpreted by Shapelets (Ye & Keogh, 2011), it may help to understand the visualization results and to prove the effectiveness of MACNN.

The CAM results of different models on the WordSynonyms dataset is shown in Fig. 8. Noted that, the darker the color is, the more contribution is offered. At first glance, the discriminative regions of time series are highlighted which means important subsequences for the classification decision, and the faded region indicates no contribution to the classification. We find that the CAM results of ResNet and FCN are almost the same since no attention mechanism is applied to these models, while SACNN highlights several discriminative regions as the result of

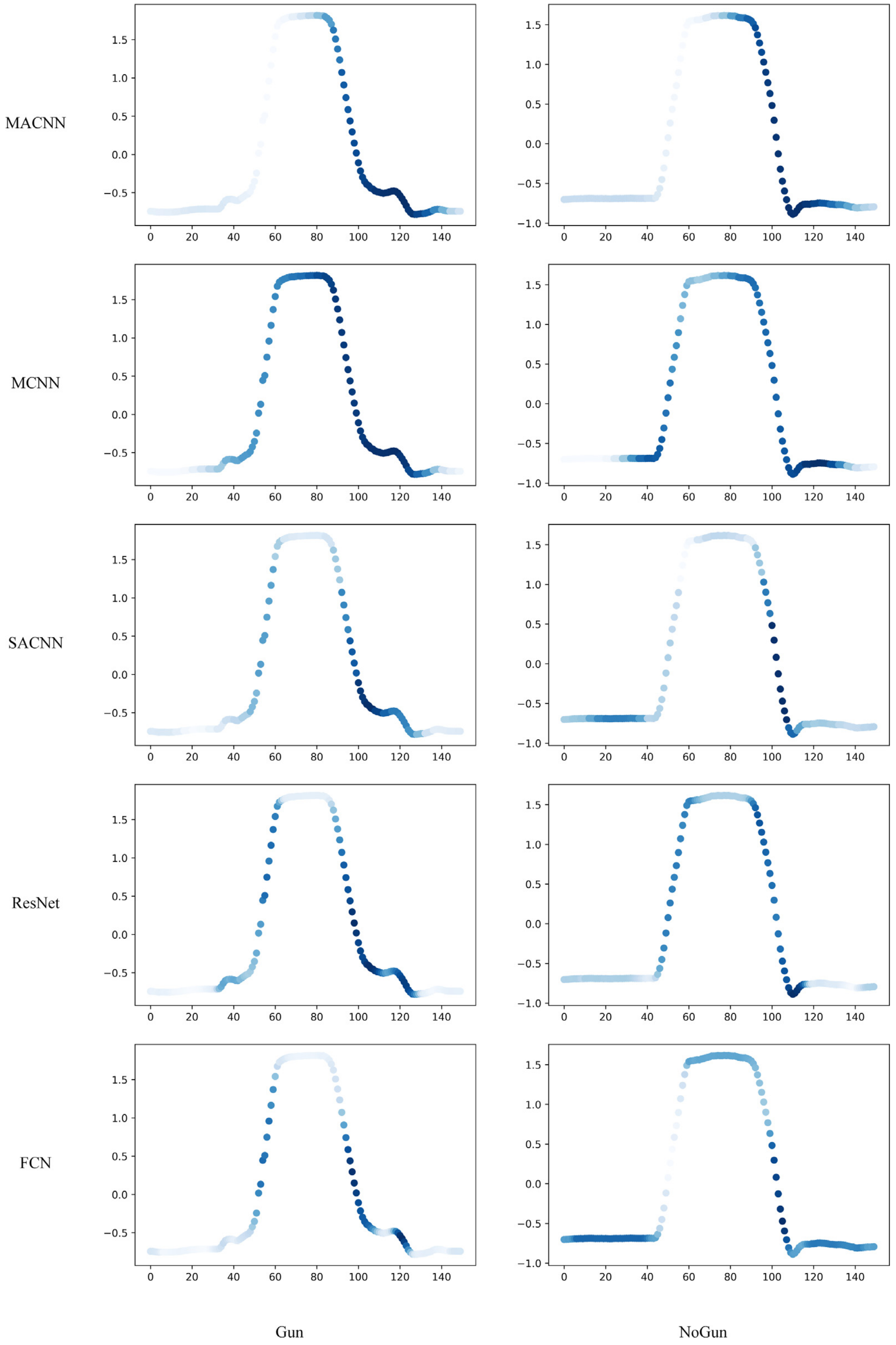


Fig. 9. CAM results of different models on the GunPoint dataset.

the attention mechanism which selectively focuses on important information. Moreover, MCNN and MACNN find out longer discriminative regions than SACNN due to the multi-scale convolution. To further understand the effectiveness of MACNN, we mainly focus on the differences between MACNN, SACNN, and MCNN. Comparing MACNN with SACNN, the discriminative regions of the two models are not quite the same, unlike SACNN which provides too many dispersed short discriminative regions, MACNN pays special attention to the most discriminative region by the combination of multi-scale convolution and attention mechanism. The multi-scale convolution captures different scales of information and the attention mechanism finds out the most important ones. In this way, MACNN takes account of the short-term, mid-term, and long-term information to achieve better performance than SACNN which captures only the single-scale of information. This indicates that introducing the multi-scale convolution is effective on TSC task. Comparing MACNN with MCNN, MCNN can divide the raw time series into a few discriminative regions by the multi-scale convolution which produces different sizes of the receptive field to capture different scales of information along the time axis. However, MACNN makes one step further to improve the classification accuracy by using attention mechanism. The attention mechanism of MACNN applies attention along channels, which enhances useful feature maps and suppresses less useful ones according to the importance of each feature map. Therefore, MACNN can select the most discriminative region more precisely compared to MCNN which includes other less important discriminative regions. This also helps to prove that the attention mechanism can improve the recognition ability of a network.

Fig. 9 shows the CAM results of MACNN, MCNN, SACNN, ResNet, and FCN models on the GunPoint dataset. Noted that the discriminative regions of all models focus on the right side of time series, which explains the same reasons for the same decision. However, there are some apparent differences between MACNN, SACNN, and MCNN. We first compare SACNN with MACNN to demonstrate the effectiveness of multi-scale convolution. As is shown in the figure, the dark area of SACNN is much shorter than MACNN, which indicates that SACNN is not able to capture a full region due to the single-scale convolution. While the multi-scale convolution produces different sizes of the receptive field to capture different scales of information, which addresses the limitation of single-scale convolution. We also compare MCNN with MACNN to explain the benefit of the attention mechanism. For MCNN, the left side of time series which is proved to not contribute to the classification is also considered to be important. On the contrary, MACNN shows that the right side of time series is the unique discriminative region which is obtained by the attention mechanism.

Another interesting proof of the effectiveness of MACNN is that the previous work by Ye and Keogh (2011) found the most discriminative shapelet for the GunPoint dataset is the "dip" area in the right bottom of the whole time series. The "dip" area of the NoGun class is explained by the "overshoot" phenomenon which indicates the actor is forced to correct her action by the inertia when she put her hand back by her side. In contrast, the Gun class has no "dip" since the actor returns her hand carefully when she holds the gun. We find that the entire "dip" area is included in the discriminative region of the CAM results of MACNN, which indicates that our discriminative region is larger and more precise to represent the whole returning action due to the combination of the multi-scale convolution and attention mechanism.

With the comprehensive comparison and evaluation above, the feasibility of deep learning based approaches for TSC task has been demonstrated. Furthermore, as a novel deep learning based architecture, MACNN achieves remarkable performance and outperforms other approaches by a large margin.

5. Conclusions and future work

In this paper, we have investigated a novel architecture for TSC task, using MACNN to recognize time series without heavy crafting in data pre-processing. We also have proposed the MAM to improve the recognition ability of a network by enhancing useful feature maps and suppressing less useful ones according to the importance of each feature map generated by the multi-scale convolution. Moreover, we have built an MA module which is the core module of the MACNN architecture to implement the MAM. We have conducted sufficient experiments and comprehensive evaluations of 85 UCR datasets compared with other competing approaches. The results show that MACNN achieves remarkable performance and outperforms other approaches by a large margin. Furthermore, we investigate the use of CAM for visualization to demonstrate the effectiveness of multi-scale convolution and attention mechanism.

In future work, we will explore deeper research to deal with some practical TSC tasks related to classification calibration and unbalanced data classification. Another important future work we are interested in is exploring ensemble method of different deep learning architectures to achieve better performance.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgment

This research was supported by the National Natural Science Foundation of China under grant No. 51435009.

References

Aswolinskiy, W., Reinhart, R. F., & Steil, J. (2017). Time series classification in reservoir- and model-space. *Neural Processing Letters*, *48*(2), 789–809.

Bagnall, A., Lines, J., Bostrom, A., Large, J., & Keogh, E. (2017). The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances. *Data Mining and Knowledge Discovery*, *31*(3), 606–660.

Bagnall, A., Lines, J., Hills, J., & Bostrom, A. (2015). Time-series classification with COTE: The collective of transformation-based ensembles. *IEEE Transactions on Knowledge and Data Engineering*, *27*(9), 2522–2535.

Bagnall, A., Lines, J., Vickers, W., & Keogh, E. (2018). The UCR time series classification repository. URL: http://www.timeseriesclassification.com/dataset.php.

Banerjee, D., Islam, K., Mei, G., Xiao, L., Zhang, G., Xu, R., Ji, S., & Li, J. (2017). A deep transfer learning approach for improved post-traumatic stress disorder diagnosis. In *Proceedings of the 2017 IEEE international conference on data mining* (pp. 11–20).

Baydogan, M. G., Runger, G., & Tuv, E. (2013). A bag-of-features framework to classify time series. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *35*(11), 2796–2802.

Demšar, J. (2006). Statistical comparisons of classifiers over multiple data sets. *Journal of Machine Learning Research*, *7*, 1–30.

Fawaz, H. I., Forestier, G., Weber, J., Idoumghar, L., & Muller, P.-A. (2019). Deep learning for time series classification: a review. *Data Mining and Knowledge Discovery*, *33*(4), 917–963.

He, K., Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. In *Proceedings of the 2016 IEEE conference on computer vision and pattern recognition* (pp. 770–778).

Hills, J., Lines, J., Baranauskas, E., Mapp, J., & Bagnall, A. (2014). Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, *28*(4), 851–881.

Hu, J., Shen, L., Albanie, S., Sun, G., & Wu, E. (2019). Squeeze-and-excitation networks. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (in press).

Hu, Q., Zhang, R., & Zhou, Y. (2016). Transfer learning for short-term wind speed prediction with deep neural networks. *Renewable Energy*, *85*, 83–95.

Iman, R. L., & Davenport, J. M. (1980). Approximations of the critical region of the friedman statistic. *Communications in Statistics*, 571–595.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: accelerating deep network training by reducing internal covariate shift. In *Proceedings of the thirty-second international conference on machine learning* (pp. 448–456).

Jaderberg, M., Simonyan, K., Zisserman, A., & Kavukcuoglu, K. (2015). Spatial transformer networks. In *Proceedings of the 2015 annual conference on neural information processing systems* (pp. 2017–2025).

Kingma, D., & Ba, J. (2015). Adam: A method for stochastic optimization. In *Proceedings of the third international conference on learning representations*.

Längkvist, M., Karlsson, L., & Loutfi, A. (2014). A review of unsupervised feature learning and deep learning for time-series modeling. *Pattern Recognition Letters*, *42*(1), 11–24.

Lin, M., Chen, Q., & Yan, S. (2014). Network in network. URL: http://arxiv.org/abs/1312.4400. unpublished results.

Lines, J., & Bagnall, A. (2015). Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, *29*(3), 565–592.

Lines, J., Taylor, S., & Bagnall, A. (2018). Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles. *ACM Transactions on Knowledge Discovery from Data*, *12*(5), 52.1–52.35.

Long, J., Shelhamer, E., & Darrell, T. (2015). Fully convolutional networks for semantic segmentation. In *Proceedings of the 2015 IEEE conference on computer vision and pattern recognition* (pp. 3431–3440).

Nair, V., & Hinton, G. (2010). Rectified linear units improve restricted Boltzmann machines. In *Proceedings of the twenty-seventh international conference on machine learning* (pp. 807–814).

Paparrizos, J., & Gravano, L. (2017). Fast and accurate time-series clustering. *ACM Transactions on Database Systems*, *42*(2), 1–49.

Rumelhart, D. E., & McClelland, J. L. (1987). Learning internal representations by error propagation. In *Parallel distributed processing: Explorations in the microstructure of cognition: Foundations* (pp. 318–362). MIT Press.

Schäfer, P. (2015). The BOSS is concerned with time series classification in the presence of noise. *Data Mining and Knowledge Discovery*, *29*(6), 1505–1530.

Schäfer, P., & Högqvist, M. (2012). SFA: A symbolic fourier approximation and index for similarity search in high dimensional datasets. In *Proceedings of the fifteenth international conference on extending database technology* (pp. 516–527).

Serrà, J., Pascual, S., & Karatzoglou, A. (2018). Towards a universal neural network encoder for time series. URL: https://arxiv.org/abs/1805.03908. unpublished results.

Szegedy, C., Liu, W., Jia, Y., Sermanet, P., Reed, S., Anguelov, D., Erhan, D., Vanhoucke, V., & Rabinovich, A. (2015). Going deeper with convolutions. In *Proceedings of the 2015 IEEE conference on computer vision and pattern recognition* (pp. 1–9).

Wang, Z., & Oates, T. (2015). Imaging time-series to improve classification and imputation. In *Proceedings of the twenty-fourth international joint conference on artificial intelligence* (pp. 3939–3945).

Wang, Z., Yan, W., & Oates, T. (2017). Time series classification from scratch with deep neural networks: A strong baseline. In *Proceedings of the 2017 international joint conference on neural networks* (pp. 1578–1585).

Woo, S., Park, J., Lee, J.-Y., & Kweon., I. S. (2018). CBAM: Convolutional block attention module. In *Proceedings of the fifteenth European conference on computer vision* (pp. 3–19).

Xi, X., Keogh, E., Shelton, C., Wei, L., & Ratanamahatana, C. A. (2006). Fast time series classification using numerosity reduction. In *Proceedings of the twenty-third international conference on machine learning* (pp. 1033–1040).

Ye, L., & Keogh, E. (2011). Time series shapelets: a novel technique that allows accurate, interpretable and fast classification. *Data Mining and Knowledge Discovery*, *22*(1–2), 149–182.

Zhou, B., Khosla, A., Lapedriza, A., Oliva, A., & Torralba, A. (2016). Learning deep features for discriminative localization. In *Proceedings of the 2016 IEEE conference on computer vision and pattern recognition* (pp. 2921–2929).