LSTM based on Adaptive Convolutional Network for Time Series Classification

Yujuan Li
Wuhan University of Technology
Yonghong Wu (whyflying2008@163.com)
Wuhan University of Technology

Research Article

Keywords: Time series classification, Adaptive selection, Long-Short-Term Memory (LSTM), Attention mechanism, Deep neural network

Posted Date: August 11th, 2022

DOI: https://doi.org/10.21203/rs.3.rs-1940259/v1

LSTM based on Adaptive Convolutional Network for Time Series Classification

Yujuan Li and Yonghong Wu

School of Science, Wuhan University of Technology, Wuhan 430070, China

Abstract: Deep learning technology is the most effective method to solve time series classification tasks. The existing algorithms based on deep learning usually obtain features with fixed step convolution, so they cannot effectively extract and focus on important multi-scale features. Based on the complexity and long-term dependence of time series data, an end-to-end model called as Adaptive Convolutional Network Long-Short-Term Memory (ACN-LSTM) is proposed in this paper. This network is composed of two branches: long-short-term memory and adaptive convolution neural network. The LSTM uses memory cells and gate mechanism to control the transmission of sequence information and fully extract the correlation information of time series to enhance the discriminative power of the network. The ACN obtains the local characteristics of time series by stacking one-dimensional convolutional neural block. Then the multi-scale convolutional neural block is used to capture different scales of information, which is consist of concatenated convolutional layers with different kernel size. Meanwhile, in order to adaptively adjust the feature information between layers, an inter-layer adaptive channel feature adjustment mechanism is proposed. The ACN-LSTM not only fully extracts long-term time correlation information, but also fuses adaptively and pays attention to important multi-scale features to achieve more accurate classification results. The experiment results with 65 UCR standard datasets illustrate that the proposed ACN-LSTM achieves highest arithmetic mean rank and geometric mean rank, compared with other methods, which are 2.815 and 2.322, respectively, and also achieves the lowest mean error with 0.127, which indicates that ACN-LSTM is effective in univariate time series classification.

Keywords: Time series classification; Adaptive selection; Long-Short-Term Memory (LSTM); Attention mechanism; Deep neural network

1.Introduction

In the era of big data, time series data widely exists in various fields including engineering, finance, medicine, human activity recognition [1]. Time series data mining has been an increasingly important research field. One of its main tasks is time series classification (TSC), where the goal is to build a classifier that can predict the category labels of time series. Since time series data is real-valued serial data, characterized by high dimensionality, variable length, multi-noise, complex relationship between variables [2]. Therefore, hundreds of methods have been done to classify time series data [3].

Conventional TSC methods mainly rely on distance similarity or features. Methods based on distance usually performs pre-defined similarity measurements on raw time series and uses existing classifiers such as KNN and SVM. The accuracy of such kind of classification methods is generally determined by the similarity measurement. One of the well-known distance-based methods is dynamic time warping (DTW) combined with KNN classifier. This method has been proved to be a very strong baseline [3]. For methods based on distance, the key part is to extract a set of representative features from a large number of labeled time series data. For example, Time series feature bag (TSBF) [4] extracts interval features by selecting random subsequences, and then applies random forest classifier to predict time series labels; BOSS [5] forms word histograms according to word distribution and uses distance of word histograms to obtain classification results. The histograms represent substructures of a time series that built on Symbolic Fourier Approximation (SFA) [6]; The fast shapelet method [7] uses symbolic approximation aggregation (SAX) [8] to convert the original sequence samples into discrete low dimensional feature subsequences, and filter out the subsequences with poor resolution in the low dimensional space. This above feature-based methods can reduce the influence of some redundant components, so that the accuracy of the classification results is further improved.

Ensemble algorithms aways combine different classifiers to obtain a higher accuracy. Elastic ensemble (PROP) [9] combines 11 elastic distance measures with 1-NN-based classifiers to construct 11 sub-classifiers, then applies the weighted voting scheme to the sub-classifiers to obtain classification results. The Collective of Transformation-Based Ensembles (COTE) [10] integrates 35 different classifiers, and performs weighted summation based on the sub-

classifiers results. Extended by this, the Hierarchical Vote Collective of Transformation-based Ensembles (Hive-COTE) [11] divides sub-classifier into different modules according to their types and then gets weighted summation based on different modules.

Although the time series classification methods have achieved remarkable results, they also have some shortcomings. For distance-based methods, they require huge feature engineering or data preprocessing. Besides, there are some shortcomings, such as low accuracy, sensitive to noise, and unable to make full use of the attributes of time series data. For feature-based approaches, the regular way is to separate the feature extraction from the classification process, which leads to the complexity of the classification. For ensemble-based approaches, it achieves relatively high classification accuracy. However, due to its high complexity, it is not widely used in practical problems.

The deep CNN is widely used in various fields [12], such as human action recognition [13] and speech emotion recognition [14], furthermore, researchers have applied it to time series classification. Fully convolutional network (FCN) [15] uses convolution layer to replace the last full connection layer of deep multilayer perceptron (MLP) [16], and adds batch standardization layer and global pooling layer to prevent the network from over fitting, which enhance the feature extraction ability of the network. Based on convolutional neural network, multi-scale convolutional neural networks (MCNN) [17] performs identity mapping, smoothing and down-sampling to extract the multi-scale features of time series data, respectively, and solve the problem of feature loss. However, its classification performance largely depends on the selection of super parameters and the quality of data preprocessing [18][19]. What's more, attention mechanism has increasingly attracted huge research attention. For example, the Multi-scale Attention Convolutional Neural Network (MACNN) [20], an end-to-end network combining multi-scale feature extraction and attention mechanism, designs a multi-scale convolution module based on convolution neural network. The network produces different ranges of receptive domains by stacking multi-scale modules to obtain the temporal feature information on different scales.

In the TSC task, time-related information is the most important features to distinguish different categories [21]. However, the networks above mentioned can only process the input time series information independently but cannot obtain the implied serial dependencies in the

time domain. In addition, most of the existing methods treat channel features equally without distinguishing the importance difference between feature channels. Under the condition of limited computational resources, if the network treats different channel features as equally important, it may result in the waste of resources. If the important features are ignored, it may seriously affect the classification performance of the network. To improve the feature extraction capability of the network, LSTM-FCN [19] has been proposed. It uses LSTM to extract correlation information of temporal data and uses FCN as a feature extraction module to extract temporal high-dimensional features. The RTFN [22] extends this method by proposing LSTM combined with CNN, jump connection structure and self-attentive mechanism to extract multi-scale temporal features.

In the TSC task, time-related information is the most important features to distinguish different categories [21]. However, the networks above mentioned can only process the input time series information independently but cannot obtain the implied serial dependencies in the time domain. In addition, most of the existing methods treat channel features equally without distinguishing the importance difference between feature channels. Under the condition of limited computational resources. Under the condition of limited computational resources, if the network treats different channel features as equally important, it may result in the waste of resources, and if the features that contribute more to the classification are ignored, it may seriously affect the classification performance of the network. To improve the feature extraction capability of the network, LSTM-FCN [19] has been proposed, which uses LSTM to extract correlation information of temporal data and uses FCN as a feature extraction module to extract temporal high-dimensional features. RTFN [22] extends this method by proposing LSTM combined with CNN, jump connection structure and self-attentive mechanism to extract multi-scale temporal features.

The approaches mentioned above also have some drawbacks as follows: 1) The existing network structure does not adequately excavate the time-related information. 2) The step size of extracted sequence features is fixed, which makes it impossible to extract multi-scale features effectively. 3) The models based on deep learning performs the same processing on different channels and do not consider the importance between feature channels, which limits the ability of feature presentation.

To solve these problems, a network ACN-LSTM based on adaptive convolution and channel attention mechanism is proposed in this paper. Its architecture focuses on two branches, one based on long-short-term memory, and the other one based on CNN named adaptive convolutional network (ACN). LSTM employs gating mechanism and memory cells to control the transmission of sequence information, which is conducive to the network learning the correlation information of time series. In ACN, stacked convolutional neural block is used to capture the local features from the input. Multi-scale block produces different sizes of the receptive field to capture different scales of information. Then the fused extracted features are introduced into the adaptive channel feature adjustment mechanism (ACFM) to learn the importance of each channel feature automatically. Finally, multi-scale features with attention weights and temporal correlation information are fused to output the classification results.

The main contributions of this paper are as follows:

(1) A time series classification network structure based on adaptive convolution and channel attention mechanism is proposed in this paper, which can enhance the accuracy classification and generalization for TSC.

(2) The adaptive channel feature adjustment mechanism (ACFM) is introduced, which automatically learns the attention weight to evaluate the importance of different channel features, and adaptively adjusts the feature information according to the channel weight.

(3) The influence of attention mechanism in adaptive channel feature adjustment mechanism is explained by using class activation map.

The rest of the paper is structured as follows. The background works are reviewed in Section 2. The architecture of the proposed model is described in Section 3. the performance of the proposed model with some comparable models is analyzed and discussed by experiments and simulation in Section 4. The main conclusions are drawn in Section 5.

2. Background works

2.1 Deep learning for TSC

In recent years, Inspired by the development of deep learning in the field of image classification, researchers are prompted to explore deep learning-based methods to solve the problems in TSC [20]. There are two kinds of deep learning algorithms for time series classification: single-network-based and dual-network-based [22]. Single-network-based algorithms usually use one network to extract features, which focus on significant important features of data. The residual neural network (ResNet) [16] identities mapping by adding shortcut to alleviate the performance degradation of deep convolution neural. Inception-Time [23] is an improved method based on GoogleNet [24], and then applied to the field of time series classification. It designs an inception module to extract potential hierarchical features of multiple time spans. OS-CNN [25] uses the full-scale module that can adapt the network parameters according to the sequence length to capture the multi-scale local features.

Dual-network-based algorithms consist of a network for extracting local features and a network for extracting the correlation between sequences. The network used for local feature extraction is usually based on convolution structure, while the network focusing on extracting the relationship between features is usually based on LSTM. As a parallel depth architecture, LSTM-FCN [19] uses LSTM to extract the correlation information and employ FCN to extract high-dimensional features. RTFN [22] uses LSTM in combination with CNN, jump connection structure and self-attention mechanism to extract multi-scale temporal features. ResNet-Transformer [26] is composed of ResNet based feature network and transformer based relational network, which solves the long-term dependency problem of the structure based on recurrent neural network. The parallel network structure algorithm usually achieves better classification performance than the single network structure algorithm [18][26].

2.2 Long-Short-Term Memory

Long-short-term memory network improves the Recurrent Neural Network (RNN) [27] with the defect of vanishing gradient problem. LSTM incorporates gating function and memory cells to control the transmission of sequence information, and can automatically store and delete the current time state information. Therefore, it can extract the complex feature

relationship of long sequences and alleviate the vanishing gradient problem in RNN. The LSTM unit is shown in Fig. 1.

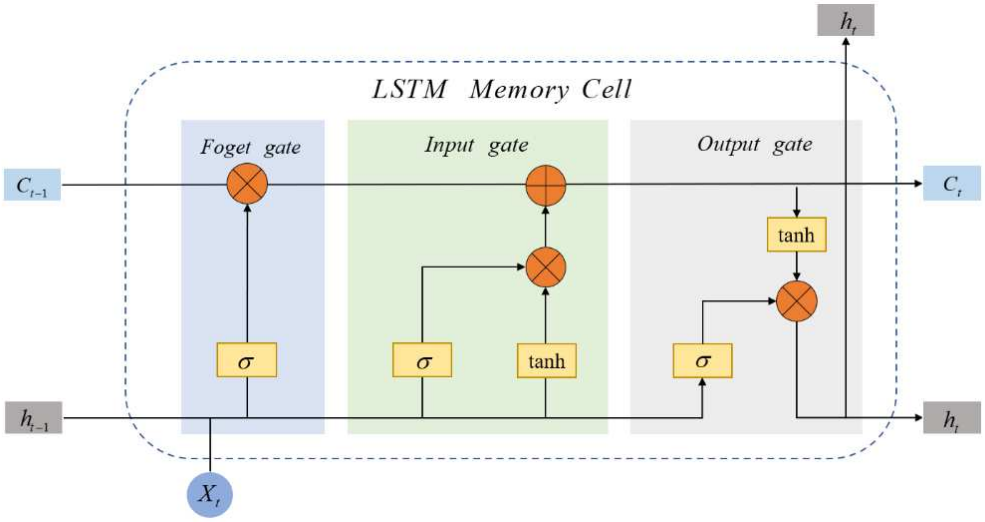

Fig. 1 Structure of LSTM unit

An LSTM unit includes a hidden vector h and a memory vector c, which is to control state updates and outputs. The computation at time step t is defined as follows [28]:

$$\boldsymbol{i}_t = \sigma(\boldsymbol{W}_{xi}\boldsymbol{x}_t + \boldsymbol{W}_{hi}\boldsymbol{h}_{t-1} + \boldsymbol{b}_i) \tag{1}$$

$$\boldsymbol{f}_t = \sigma(\boldsymbol{W}_{xf}\boldsymbol{x}_t + \boldsymbol{W}_{hf}\boldsymbol{h}_{t-1} + \boldsymbol{b}_f) \tag{2}$$

$$\boldsymbol{o}_t = \sigma(\boldsymbol{W}_{xo}\boldsymbol{x}_t + \boldsymbol{W}_{ho}\boldsymbol{h}_{t-1} + \boldsymbol{b}_o) \tag{3}$$

$$\tilde{\boldsymbol{c}}_t = tanh(\boldsymbol{W}_{xc}\boldsymbol{x}_t + \boldsymbol{W}_{hc}\boldsymbol{h}_{t-1} + \boldsymbol{b}_c) \tag{4}$$

$$\boldsymbol{c}_t = \boldsymbol{f}_t \text{ e } \boldsymbol{c}_{t-1} + \boldsymbol{i}_t \text{ e } \tilde{\boldsymbol{c}}_t \tag{5}$$

$$\boldsymbol{h}_t = tanh(\boldsymbol{o}_t \cdot \boldsymbol{c}_t) \tag{6}$$

where x_t denotes the information input at the time t, c denotes the cell state, saving the sequence information extracted by the network; i denotes an input gate, which controls the amount of information input from the current x_t to the memory cell; f denotes forget gate, which controls the amount of information from c_{t-1} to c_t. o denotes output gate that controls the amount of information that c_t transmitted to the hidden layer h_t; W_{hi}, W_{hf}, W_{ho}, W_{hc} are recurrent weight matrices, W_{xi}, W_{xf}, W_{xo}, W_{xc} are projection matrices and

b_i, b_f, b_o, b_c are the bias vectors. $\sigma(\cdot)$ indicates the sigmoid function; $tanh(\cdot)$ denotes hyperbolic tangent function; e denotes Hadamard product.

2.3 Attention Mechanism

The attention mechanism is a tool to enhance the presentation ability of the network by focusing on important features and suppressing unnecessary features. So that, it can efficiently allocate computing resources to obtain the most important feature presentation [29][30]. The current attention mechanism can be roughly classified into spatial attention and channel attention. Spatial Transformer Networks (STN) [31] is a spatial attention network, which transforms the spatial information of the original sample to another space and retains its key information, so that the network can automatically select the most relevant area of the image and realize the extraction of local important information of the sample.

Channel attention mechanism has been applied to some network models, such as SE network [32]. It integrates the information of the channel dimension, obtains the importance weight of each feature channel automatically. Then the generated weight is multiplied by the original feature channel one by one, so as to achieve the purpose of "feature recalibration". Moreover, SK network [33] is a dynamic selection mechanism for convolution kernel in CNN, which enables the network to adaptively adjust the size of receptive field according to the input information. Convolutional Block Attention Module (CBAM) [34] applies channel and spatial attention module to obtain attention map, and then multiplies the attention map with the input feature map for adaptive feature optimization.

3. Method

3.1 Overview architecture

The structure of the ACN-LSTM consists of LSTM-based network and adaptive convolutional network, as shown in Fig. 2. The ACN branch is used to extract the depth local feature information of the network, and the LSTM-based branch is used to extract the relation information in time series data. Then, the feature joint layer is used to connect the multi-scale attention features and correlation information extracted by the two branches, input SoftMax to calculate the posterior probability of each category, and output the classification results. Supervision is provided at the output layer of the network, and the network parameters are learned according to the cross-entropy loss function.

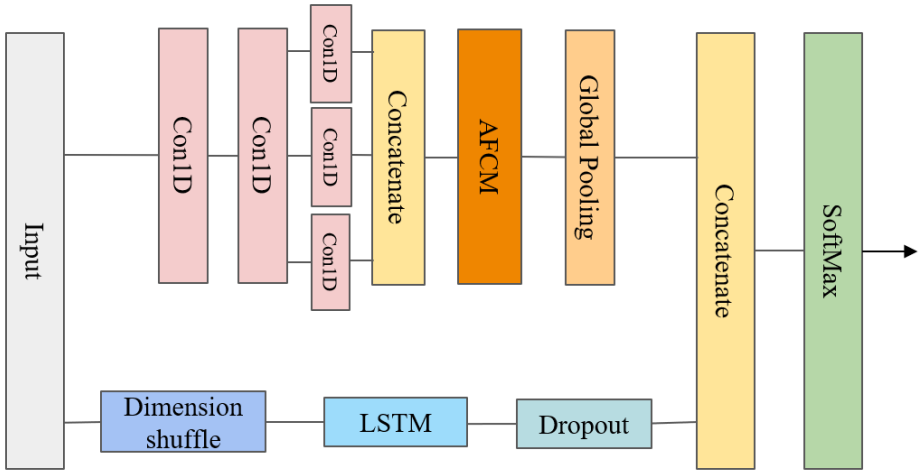

Fig. 2 Structure of the ACN-LSTM

3.2 LSTM-based network

In LSTM-based network, the dimension shuffle layer converts univariate time series samples with length of N into N-dimensional multivariable time series with a single time step. Therefore, it avoids over fitting for short series and the acquisition of long-term correlation for long series. The dimension shuffle operation can significantly improve the training speed of LSTM [19].

LSTM controls the transmission of information and extracts time-related information. Then it inputs the extracted features into the dropout layer [35], and removes neurons according to a certain probability p during network training, which can prevent over fitting of the model

and increase the robustness of the model. After the dropout operation is added, the network calculation formula is as follows:

$$r_j^{(l)} \sim \text{Bernoulli}(p) \tag{7}$$

$$\overset{\circ}{\boldsymbol{x}}^{(l)} = r^{(l)} * \boldsymbol{x}^{(l)} \tag{8}$$

$$z_i^{(l+1)} = \boldsymbol{w}_i^{(l+1)} \overset{\circ}{\boldsymbol{x}}^{(l)} + b_i^{(l+1)} \tag{9}$$

$$y_i^{(l+1)} = f\left(z_i^{(l+1)}\right) \tag{10}$$

where Bernoulli denotes Bernoulli distribution. Parameters p denote the probability of removing neurons. $r_j^{(l)}$ take the probability p and $1-p$ to get 1 and 0, respectively, which is used to adjust the input $\boldsymbol{x}$ of the neural network. $\boldsymbol{w}_i^{(l+1)}$ and $b_i^{(l+1)}$ denote the weight vector and bias term of the ith neuron on the layer $l+1$, respectively. $f(\cdot)$ is the Rectified Linear Unit activation function.

3.3 Adaptive convolutional network

The subsequence of time-series data contains complex information, and the loss of subsequence characteristics of the network often causes data misclassification. ACN is used for depth local feature information extraction and multi-scale features. There are two convolutional neural blocks to obtain local important features from the input data and extract deeper information; One multi-scale convolutional neural block is used to extract the multi-scale features of the network. Adaptive channel feature adjustment mechanism (ACFM) is used to enable the network to adaptively adjust the size of convolution kernel according to the length of input.

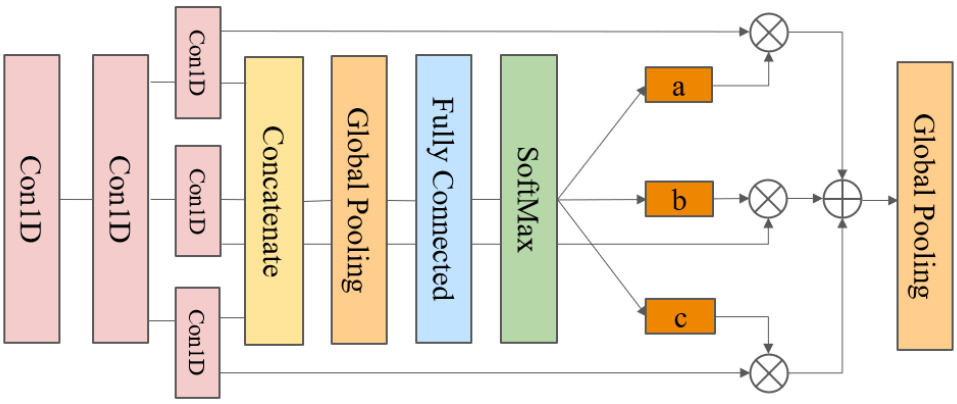

Fig. 3 Adaptive Convolutional Network

The structure of the adaptive convolutional network is shown in Fig.3. The first two layers of the network are one-dimensional Convolutional Neural block, namely "Conv1D", which can obtain local important features from the input and extract deeper information by stacking. It uses the multi-scale convolutional neural block to extract the multi-scale features, fuse more abundant local information, and make full use of the hidden information in the sequence. Finally, an inter channel adaptive feature adjustment mechanism (ACFM) is embedded between layers. It learns the depth presentation of features from shallow features, and adaptively adjusts feature information by using time information and channel correlation.

3.2.1 Convolutional Neural Block (Conv1D)

A Conv1D block consists of a 1-dimensional CNN module, batch normalization module [36], and ReLU activation function [37]. $a_{i,t}^{(l)}$ denotes the eigenvalue obtained by CNN on the ith filter at time t, $f_{B\&N}(\cdot)$ is the activation value by batch normalization as follows:

$$a_{i,t}^{(l)} = f_{conv}\left(\sum_{t=1}^{d}\left\langle W_{i,t',\cdot}^{(l)}, x_{\cdot,t+d-t'}^{(l-1)}\right\rangle + b_i^{(l)}\right) \tag{11}$$

$$f_{B\&N}\left(\boldsymbol{a}^{(l)}\right) = \frac{\boldsymbol{a}^{(l)} - E\left[\boldsymbol{a}^{(l)}\right]}{\sqrt{Var\left[\boldsymbol{a}^{(l)}\right]}} \tag{12}$$

where $W^{(l)} \in R^{F_l \times d \times F_{l-1}}$ denotes the weight tensor of CNN at the layer l, $b^{(l)} \in R^{F_l}$ denotes the bias, d denotes the size of convolution kernel, and E denotes the activation value.

Then it is introduced into the activation layer, and the rectified linear unit (ReLU) is used as the activation function, which can enhance the nonlinear presentation relationship of the network and produce a relatively sparse learning parameter matrix to reduce the computational complexity of the network. To improve the performance and robustness of the model, combined with BN operation and ReLU activation function, the output of one-dimensional convolution module is as follows:

$$f_{conv1D} = f_{\mathrm{Re}LU}(f_{B\&N}(f_{conv}(x))) \tag{13}$$

3.2.2 Multi-scale convolutional neural block

Multi-scale convolution is to extract the multiscale features of sequences by using three convolution kernels with different scales on the basis of convolution neural network. In the multi-scale convolution module, set the convolution kernel of three steps for CNN, set the step size, and extract the short, medium and long-term features of time series correspondingly. Input multi-scale features into the concatenate layer for feature fusion, combined with BN operation and ReLU activation function, the output is:

$$\boldsymbol{Y} = f_{\mathrm{Re}LU}(f_{B\&N}(f_{concat}([O_{\mathrm{short}}, O_{\mathrm{midium}}, O_{\mathrm{long}}]))) \tag{14}$$

where $O_{shart}, O_{medium}, O_{long}$ denote the feature map extracted by the convolution kernel size d_1, d_2, d_3, respectively; $f_{concat}(\cdot)$ denotes the feature connection operation.

3.2.3 Adaptive channel feature adjustment mechanism

Inspired by the application of channel attention mechanism in computer vision [38], the adaptive channel feature adjustment mechanism (ACFM) is proposed in this paper, which learns the depth presentation of features from shallow features and uses time information and channel correlation to automatically correct features, such that the network can adaptively allocate the size of convolution kernel according to the sequence length to extract multi-scale features. The structure of ACFM is shown in Fig. 4. It uses the average pooling layer to obtain the global information of all channels, then reduces the number of channels, fuses the feature information between channels, calculates the channel attention weights of different convolution cores through the SoftMax function, and weights the generated weights with the original feature channels channel by channel to achieve the purpose of "feature recalibration", enhance useful features and suppress less useful features.

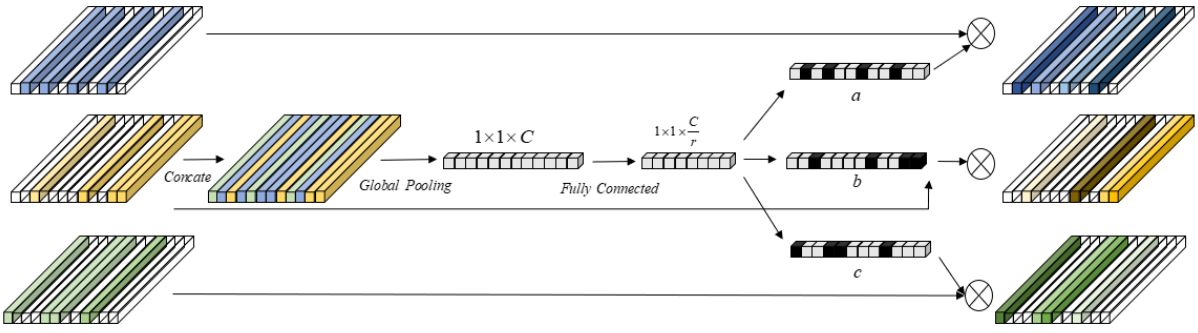

Fig. 4 Detail of the adaptive feature adjustment mechanism

First, the feature output by the multi-scale convolution module is used as the input of ACFM for global pooling operation to obtain the global description $\tilde{Y} \in R^{c\times 1}$:

$$\tilde{\boldsymbol{Y}} = f_{avg\_pool}(\boldsymbol{Y}) \tag{15}$$

$$\tilde{y}_n = \frac{1}{L_f} \cdot \sum_{i=1}^{L_f} y_n(i) \tag{16}$$

where $f_{avg\_pool}(\cdot)$ denotes global average pooling operation, which fuses the global information of the time series feature map and generates the global distribution of channel features, such that the network can use the information extracted from the global perception domain. The characteristic graph $\boldsymbol{Y} \in \mathbf{R}^{L_f \times C}$ is transformed into a real valued vector matrix $\tilde{\boldsymbol{Y}} \in \mathbf{R}^{1\times C}$.

Then pass it into the full connect layer, and use the ReLU function as the activation function. In the full connection layer, the number of characteristic channels is compressed into input $1/r$ to reduce the dimension, r denotes the dimension reduction factor. The full connection layer fuses the global feature information of the channels, and uses the global information to obtain the nonlinear correlation between the channels. The operation is as follows:

$$z = f_{\mathrm{ReLU}}(W\tilde{Y}) \tag{17}$$

where f_{ReLU} is ReLU function, $\boldsymbol{W} \in R^{\frac{C}{r}\times C}$ denotes the weight coefficient of neurons in the full connection layer.

The channel attention weight a_c, b_c, c_c adaptively select the characteristic information of different scales. The channel attention weight corresponding to each convolution block is obtained through SoftMax as follows:

$$a_c = \frac{e^{A_c z}}{e^{A_c z} + e^{B_c z} + e^{C_c z}}, b_c = \frac{e^{B_c z}}{e^{A_c z} + e^{B_c z} + e^{C_c z}}, c_c = \frac{e^{C_c z}}{e^{A_c z} + e^{B_c z} + e^{C_c z}} \tag{18}$$

where $A, B, C \in R^{C\times d}$, a, b, c donate the attention weight coefficients of three convolution kernels with different scales, respectively.

Finally, multiply the three features output $O_{\mathrm{short}}, O_{\mathrm{midium}}, O_{\mathrm{long}}$ by the multiscale convolution

module and the weight output a_c, b_c, c_c by the corresponding attention module element by element to obtain the output f of ACFM, and input it into the subsequent network layer. The calculation formula of characteristic attention weight is as follows:

$$F = a_c \cdot O_{short} + b_c \cdot O_{medium} + c_c \cdot O_{long}, a_c + b_c + c_c = 1 \quad (19)$$

If the feature is more important, its corresponding attention weight is closer to 1; On the contrary, the attention weight is closer to 0. The network can judge the importance of features according to the weight, allocate more computing resources for important features, and suppress unimportant features to improve the quality of feature presentation generated by the network.

4. Experiments

This section introduces the datasets and evaluation indicators used in the experiment, shows the selection of network architecture, network parameter settings, and describes the relevant settings of the experiment. Finally, it makes a comparative experiment with 9 most advanced time series classification models.

4.1 Dataset and Evaluation

4.1.1 Dataset

ACN-LSTM uses 65 UCR univariate time series datasets used by Xiao et al. [22] in the comparative experiment, involving six fields: ECG, image, motion, sensor, simulated, and Spectro [39]. The sequence length ranges from 24 to 2709, covering short, medium and long sequence datasets. The number of samples in the dataset ranges from 20 to 4500, and the number of categories between 2 and 60. All datasets have been normalized to zero mean and unit variance, thus, there is no additional data preprocessing. These datasets are presentative of real time series data, and the experimental results can prove the performance and generalization ability of the model.

4.1.2 Evaluation

In this paper, three time series classification evaluation indicators are selected to evaluate each model, and Wilcoxon signed rank test and Nemenyi test are used to compare the performance differences between this model and other models. The definitions are as follows:

（1）Arithmetic Mean Rank (AMR) refers to the arithmetic mean value of the accuracy ranking of the target model on all datasets. The lower the value, the better the comprehensive performance of the model. The calculation formula is as follows:

$$E_{\mathrm{AMR}} = \frac{1}{N}\sum_{i=1}^{N} r_i \tag{20}$$

（2）Geometric Mean Rank (GMR) denotes the geometric average of the accuracy ranking of the target model on all datasets, and measures the comprehensive performance of the model on the datasets together with AMR. If a model has a large GMR value, it means that the accuracy of the model ranks low on multiple datasets. The calculation formula is as follows:

$$E_{\mathrm{GMR}} = \sqrt[N]{\prod_{i=1}^{N} r_i} \tag{21}$$

（3）Mean Error (ME) indicates the accuracy of the target model on all datasets. The lower the value, the better the model performance. The calculation formula is:

$$E_{\mathrm{ME}} = \frac{1}{N}\left(\sum_{i=1}^{N}(1-a_i)\right) \tag{22}$$

where E denotes the evaluation index, a_i, r_i denotes the top-1 accuracy and accuracy ranking of the target model on the dataset, and N denotes the total number of datasets.

（4）Wilcoxon signed-rank test (WST) is a nonparametric statistical test, which compares the accuracy ranking of other models on each dataset to test whether the performance of this model is significantly better than that of other comparison models. Its original and alternative assumptions are as follows:

$$H_0 : M_{ACN-LSTM} = M_{\mathrm{model}_i}$$
$$H_1 : M_{ACN-LSTM} \neq M_{\mathrm{model}_i}$$

M denotes the median of the accuracy of the model on all datasets, and i denotes the *ith* other model.

（5）Friedman test is a nonparametric test that uses rank to test whether there is a significant difference in the accuracy of multiple models. The original and alternative assumptions are as follows:

H_0 : There is no significant difference in the accuracy of all models

H_1 : There is a significant difference in the accuracy of all models

The variable Friedman statistic F is calculated as follows:

$$\chi^2 = \frac{12N}{k(k+1)}\sum_{i=1}^{k} r_i^2 - 3N(k+1) \tag{23}$$

$$F = \frac{(N-1)\chi^2}{N(k-1)-\chi^2} \tag{24}$$

where k and N refer to the number of methods and datasets, respectively; r_i refers to the value of AMR of *ith* model, which r_i is distributed according to the F distribution with $\frac{k+1}{2}$ and $\frac{k^2-1}{12}$ degrees of freedom.

（6）Nemenyi test. If the assumption that there is no significant difference in the accuracy of all models is rejected, it indicates that there is a significant difference in performance between models. The Nemenyi follow-up test can further distinguish the advantages and disadvantages of models. Calculate the critical difference (CD) according to the corresponding confidence, and the calculation formula is as follows:

$$CD = q_a \sqrt{\frac{k(k+1)}{6N}} \tag{25}$$

where CD denotes the critical difference, k denotes the number of models, N denotes the total number of datasets, and q_α is the critical value of Tukey distribution. Under the experimental conditions in this paper, the confidence level $\alpha = 0.05$, $q_\alpha = 3.102$. If the difference between the average rankings of the two algorithms exceeds the critical difference, the assumption that "there is a significant difference in the accuracy of two models" is rejected.

4.2 Experiment settings

In this paper, 7 time series datasets with different sequence lengths involving multiple fields from UCR are used to choose optimal parameters. The details of these 7 datasets are shown in Tab. 1.

Tab. 1 Information of 8 time series datasets

Name	Class	Length	Train	Test
Chlorine	3	166	467	3840
MoteStrain	2	84	20	1252
CinCECG	4	1639	40	1380
CricketX	12	300	390	390
FacesUCR	14	131	200	2050
ItalyPower	2	24	67	1029
MALLAT	8	1024	55	2345

The ACN branch of ACN-LSTM can extract important multi-scale time series features. However, due to the unequal length of the sample sequence of the dataset used in this paper, it is necessary to set up a suitable sensing domain to adapt to most datasets when extracting short,

medium and long-term features. Therefore, the core size of the first convolution layer of the ACN branch is set to 8 and the number of convolution cores is set to 64; The core size of the second convolution layer is set to 5, and the number of convolution cores is 128; The multiscale convolution kernel in ACN is fixed to 3, 5 and 11, respectively, and the convolution sliding step is fixed to 1 in [19][20][32].

To further determine the optimal hyperparameter combination of the setup model, the hyperparameter search method and initialization hyperparameter combination in [32] are used in this paper: first, set the initial values of batch size, number of training rounds, dimension reduction factor and rejection rate as [32, 2000, 16, 0.8]. Then the greedy strategy based on the combination of super parameters is used to search the optimal parameter value. Specifically, the last three parameters are fixed, the different values of the first parameter are tested, and the value that makes the model reach the lowest error rate is selected as the optimal value of the parameter. Then fix the first, third and fourth parameters, select the optimal value of the second parameter, and so on. The parameters with the optimal value are fixed as the optimal value in the next parameter search experiment.

Set the network parameter search space according to [19][20], and select the batch size, rejection rate, dimension reduction factor and number of training rounds in [32, 64, 128, 256], [0.5, 0.6, 0.7, 0.8, 0.9], [8, 16, 32] and [1000, 1500, 2000, 2500], respectively. Then, by using 3x cross validation and manually changing the value of the hyperparameter, the performance of different combinations of hyperparameters is evaluated. Tab. 2 shows the average error rate of each network parameter combination.

Tab. 2 Mean error of each parameter set

Batch size	Training epoch	Reduction factor	dropout	Averaged validation error
32	1500	16	0.8	0.145
64	1500	16	0.8	0.142
128	1500	16	0.8	**0.131**
256	1500	16	0.8	0.139
128	1000	16	0.8	0.138
128	1500	16	0.8	0.131

128	**2000**	16	0.8	**0.131**
128	2500	16	0.8	**0.131**
128	2000	8	0.8	0.132
128	2000	**16**	0.8	**0.131**
128	2000	32	0.8	0.134
128	2000	16	0.5	0.136
128	2000	16	0.6	0.134
128	2000	16	0.7	0.136
128	2000	16	**0.8**	**0.131**
128	2000	16	0.9	0.134

According to the experimental results of parameter selection in Tab. 2. It is found that in the given network parameter space, the minimum average error rate of the model is 0.131. Among them, the obtained performance of the model is the same when the number of training rounds is 2000 and 2500. Under the principle of saving computing resources, 2000 is chosen as the optimal value of the number of training rounds.

The parameter selection of the model is as follows: the batch size is set to 128, the initial learning rate is set to 1e-3, the epoch is set to 2000, and the Adam optimizer [39] is used for training. If the accuracy of the model on the test set is not improved for 100 consecutive rounds, the learning rate will be reduced to the original $1/\sqrt[3]{2}$ one until it is reduced to 1e-4. The size of the three convolution cores of ACN is set to 3, 5 and 11, respectively; the convolution sliding step is set to 1, and the number of convolution cores is set to 128. The dimension reduction factor in the attention module is set to 16. The number of memory cells of LSTM is selected from 8, 64, 128, and the deletion rate of deletion layer is set to 0.8.

4.3 Experiments Results and Analysis

The code of ACN-LSTM model is written using the deep learning framework Keras [40] and trained on NVIDIA GeForce GTX 1650 GPU. Since the neural network uses random initial weights, 10 experiments are carried out to average the error caused by the initial weights.

Conduct experiments on 65 datasets in UCR, and compare them with 9 methods: Distance-based methods: DTW[4]; Feature-based methods: TSBF[6], BOSS[7]; ensemble-based

methods: PROP[12], COTE[13]; Neural network methods: FCN [17], ResNet[22], OS-CNN[26], Inception-Time[28]。

4.3.1 Top-1 accuracy

To evaluate the performance of ACN-LSTM and other models on all datasets, calculate the arithmetic mean rank (AMR), geometric mean rank (GMR) and average error rate (me) of evaluation indicators according to (20), (21) and (22), evaluate the performance of each method, and use “best” to denote the number of models that obtain the highest accuracy on the dataset. Use Wilcoxon signed rank test (WST) to measure whether there is a significant difference between the accuracy of ACN-LSTM and other models. AMR meets the conditions of Friedman test, thus, Friedman test and Nemenyi test are used to compare the accuracy performance of all models. Tab. 3 shows the top-1 accuracy and the values of 4 evaluation indicators obtained by ACN-LSTM and 9 comparison models on 65 univariate time series datasets.

Tab. 3 The comparison of Top-1 accuracy rates on 65 datasets

Dataset	DTW	TSBF	BOSS	PROP	COTE	FCN	ResNet	OS-CNN	Inception-Time	ACN-LSTM
Adiac	0.604	0.769	0.764	0.664	0.79	**0.844**	0.826	0.838	0.841	0.808
ArrowHead	0.663	0.754	0.834	0.811	0.811	0.843	0.817	0.84	0.845	**0.897**
Beef	0.633	0.566	0.8	0.633	**0.866**	0.697	0.767	0.833	0.7	0.767
BeetleFly	0.7	0.8	**0.9**	0.75	0.8	0.86	0.8	0.8	0.8	0.85
BirdChicken	0.75	0.9	**0.95**	0.8	0.9	**0.95**	0.9	0.9	**0.95**	**0.95**
Car	0.733	0.783	0.833	0.833	0.9	0.905	**0.933**	**0.933**	0.883	**0.933**
CBF	0.997	0.987	0.997	0.997	0.995	0.994	0.994	0.998	0.998	**0.999**
ChlorineConcentration	0.648	0.692	0.66	0.656	0.727	0.814	0.828	0.849	**0.876**	0.828
CinCECGTorso	0.651	0.712	0.886	0.942	**0.994**	0.824	0.826	0.83	0.853	0.896
Coffee	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
Computers	0.7	0.756	0.756	0.708	0.740	**0.822**	0.815	**0.822**	0.807	0.813
CricketX	0.754	0.705	0.735	0.812	0.807	0.792	0.821	0.846	**0.853**	0.813
CricketY	0.744	0.735	0.753	0.805	0.825	0.787	0.805	**0.869**	0.851	0.826

CricketZ	0.754	0.715	0.746	0.782	0.815	0.811	0.813	**0.861**	**0.861**	0.823
DiatomSizeReduction	0.967	0.898	0.931	0.944	0.928	0.687	0.931	**0.98**	0.934	0.928
DistalPhalanxAgeGroup	0.77	0.782	0.77	0.728	0.76	0.71	0.798	0.755	0.733	**0.823**
DistalPhalanxCorrect	0.717	0.712	0.739	0.69	0.748	0.76	0.771	0.771	0.782	**0.798**
Earthquakes	0.719	0.748	0.748	0.741	0.748	0.727	**0.786**	0.683	0.741	0.78
ECG200	0.77	0.84	0.87	0.88	0.88	0.889	0.87	0.91	**0.93**	0.92
ECG5000	0.75	0.939	0.941	0.938	**0.946**	0.940	0.931	0.94	0.94	0.945
ECGFiveDays	0.768	0.876	1	0.819	0.998	0.987	0.955	**1**	**1**	**1**
ElectricDevices	0.601	0.703	0.725	0.663	0.713	0.702	0.718	0.718	**0.729**	0.727
FaceUCR	0.83	**1**	1	0.909	0.897	0.928	0.932	0.943	0.954	0.955
FordA	0.555	0.85	0.914	0.737	0.956	0.904	0.928	0.958	**0.961**	0.93
FordB	0.62	0.598	0.82	0.661	0.803	0.878	**0.9**	0.813	0.861	0.897
Gun_Point	0.907	0.986	**1**	0.993	**1**	**1**	0.993	**1**	0.987	**1**
Ham	0.467	0.761	0.666	0.571	0.647	0.718	**0.781**	0.714	0.714	0.733
HandOutlines	0.881	0.854	0.911	0.889	0.918	0.806	0.861	0.956	0.954	**0.957**
Haptics	0.377	0.49	0.461	0.392	0.522	0.48	0.506	0.512	**0.548**	0.494
Herring	0.531	0.64	0.546	0.578	0.625	0.608	0.594	0.609	0.671	**0.734**
ItalyPowerDemand	0.95	0.883	0.908	0.962	0.961	0.961	0.96	0.947	0.965	**0.968**
Lightning2	0.869	0.737	0.836	**0.885**	0.868	0.739	0.754	0.819	0.77	0.803
Lightning7	0.726	0.726	0.684	0.767	0.808	0.827	0.836	0.808	0.835	**0.849**
MALLAT	0.934	0.96	0.938	0.939	0.953	0.967	**0.979**	0.963	0.955	0.961
Meat	0.933	0.933	0.9	0.933	0.916	0.853	**1**	0.983	0.933	0.933
MedicalImages	0.737	0.705	0.718	0.742	0.757	0.779	0.772	0.768	0.794	**0.814**
MiddlePhalanxAgeGroup	0.5	**0.814**	0.565	0.783	0.804	0.553	0.76	0.538	0.551	0.728
MiddlePhalanxCorrect	0.698	0.577	0.77	0.558	0.636	0.801	0.793	0.807	0.817	**0.837**
MiddlePhalanxTW	0.506	0.597	0.526	0.512	0.571	0.512	**0.607**	0.564	0.512	0.604
MoteStrain	0.835	0.903	0.878	0.882	0.936	0.937	0.895	**0.939**	0.886	0.916
OliveOil	0.833	0.833	0.866	0.866	**0.9**	0.723	0.867	0.833	0.833	0.833
Plane	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**

ProximalPhalanxAgeGroup	0.805	**0.872**	0.848	0.807	0.869	0.831	0.849	0.843	0.848	0.850
ProximalPhalanxCorrect	0.784	0.848	0.834	0.804	0.853	0.903	0.918	0.9	**0.931**	0.911
ProximalPhalanxTW	0.756	0.809	0.815	0.765	0.78	0.767	0.807	0.775	0.775	**0.9**
RefrigerationDevices	0.564	0.472	0.725	0.676	**0.742**	0.533	0.528	0.558	0.564	0.592
ScreenType	0.397	0.50933	0.586	0.554	0.651	0.667	**0.707**	0.661	0.657	0.509
ShapeletSim	0.65	0.961	**1**	0.816	0.961	0.724	**1**	0.827	0.955	0.989
ShapesAll	0.768	0.185	0.908	0.866	0.891	0.895	0.912	0.923	0.928	**0.961**
SonyAIBORobot1	0.725	0.795	0.632	0.703	0.845	0.96	**0.985**	0.978	0.868	0.953
SonyAIBORobot2	0.831	0.777	0.859	0.878	0.951	**0.979**	0.962	0.961	0.946	0.974
Strawberry	0.941	0.954	0.976	0.945	0.951	0.972	0.958	0.981	**0.983**	0.974
SwedishLeaf	0.792	0.915	0.921	0.915	0.955	0.969	0.958	0.969	**0.977**	0.962
Symbols	0.95	0.945	0.966	0.959	0.963	0.955	0.872	0.976	0.98	**0.984**
SyntheticControl	0.993	0.993	0.966	**1**	**1**	0.985	**1**	**1**	0.996	**1**
ToeSegmentation1	0.772	0.78	0.938	0.828	0.973	0.961	0.965	0.956	0.964	**0.997**
ToeSegmentation2	0.838	0.8	0.961	0.892	0.915	0.88	0.862	0.938	0.938	**0.965**
Trace	**1**	0.98	**1**	0.99	**1**	**1**	**1**	**1**	**1**	**1**
TwoLeadECG	0.904	0.976	0.993	**1**	**1**	0.871	**1**	0.999	0.995	**1**
UWaveX	0.728	0.83	0.762	0.805	0.821	0.817	0.787	0.817	0.824	**0.843**
UWaveY	0.634	0.736	0.685	0.725	0.758	0.754	0.668	0.749	0.767	**0.778**
UWaveZ	0.658	0.772	0.694	0.723	0.75	0.639	0.755	0.757	0.764	**0.789**
UWaveAll	0.892	0.926	0.938	**0.968**	0.964	0.726	0.868	0.941	0.951	0.966
Wafer	0.98	0.995	0.994	0.997	**0.999**	0.997	0.997	0.998	0.998	**0.999**
Wine	0.574	0.611	0.74	0.574	0.648	0.587	0.796	0.555	0.611	**0.833**
WordSynonyms	0.649	0.688	0.637	**0.774**	0.757	0.564	0.632	0.747	0.733	0.759
ME	0.249	0.207	0.173	0.196	0.150	0.176	0.149	0.147	0.146	**0.127**
AMR	8.592	7.046	5.992	6.862	4.815	5.869	4.792	4.254	3.962	**2.815**
GMR	8.238	6.266	5.325	6.184	4.270	5.148	3.978	3.714	3.303	**2.322**
Best	3	5	9	7	13	8	15	12	16	**29**

Compared with 9 temporal classification methods, the ACN-LSTM model in this paper

achieves the highest top-1 accuracy on 29 datasets. Moreover, the me, AMR and GMR evaluation indexes obtained from 65 UCR univariate time series datasets are better than the other 9 models. The specific values of the three evaluation indicators are 0.127, 2.815 and 2.322, respectively. Compared with other model architectures for extracting multi-scale feature classification, for example, inception time extracts multi-scale features by stacking CNN based feature extraction modules to generate different receptive domains, and OS-CNN adaptively allocates convolution kernel size to extract multi-scale features according to sequence length. Compared with inception time, ACN-LSTM reduced me index by 1.9%, and AMR and GMR increased by 1.147 and 0.981, respectively; Compared with OS-CNN model, me decreased by 2%, and AMR and GMR increased by 1.439 and 1.392, respectively. Due to the fact that ACN-LSTM combines the attention mechanism on the basis of multi-scale convolution, the model is more effective in fusing multi-scale information and paying attention to important features.

For more comparisons, the number of datasets with better, the same or worse performance than ACN-LSTM is counted. When the average accuracy of model A is 0.01 higher or lower than that of model B, we judge that model A is significantly better or worse than model B in the dataset. In this paper, Wilcoxon signed rank test (WST) is used to measure the significance of the difference between ACN-LSTM and the other nine methods. The significance p value is shown in Tab. 4. If the p value of the test is less than 0.05, the original hypothesis is rejected, which indicates that the median of the top-1 accuracy ranking of the two models on all datasets is not equal, that is, the classification performance of the two models is significantly different. On the contrary, if the original hypothesis is accepted, the performance of the two models is similar.

Tab. 4 Pairwise comparison of all models against ACN-LSTM

Length	DTW	TSBF	BOSS	PROP	COTE	FCN	ResNet	OS-CNN	Inception-Time
Worse	2	4	7	7	10	3	9	14	10
Better	57	51	42	45	36	45	32	31	33
tie	6	10	16	13	19	17	24	20	22
p(WST)	3.44E-11	7.98E-10	2.63E-07	2.45E-08	2.95E-04	2.67E-08	1.13E-04	0.006729	7.21E-04

Tab. 4 shows ACN-LSTM achieves higher accuracy on most datasets compared with the time series classification models. Compared with OS-CNN and Inception-Time, which also use multi-scale convolution, ACN-LSTM has achieved better performance on 31 and 33 datasets, respectively, and the accuracy of the two multi-scale models is the same on 20 and 22 datasets, respectively. Tab. 4 shows the p value corresponding to Wilcoxon signed rank test, which proves that ACN-LSTM is the most accurate classifier and achieves better performance than other methods.

To further evaluate the performance of all methods, we use Friedman test to compare multiple classifiers. When $\alpha = 0.05$, the critical value of F (9576) is 1.896, and the value of F calculated according to formulas (23) and (24) is 29.920>1.896. Therefore, the original assumption that "there is no significant difference in the accuracy of all models" is rejected. Then calculate the critical difference according to (25) for Nemenyi subsequent test to further judge the performance of each model. According to the critical difference diagram between the models shown in Fig.5, it is concluded that the overall classification performance of ACN-LSTM is significantly better than ResNet, COTE, FCN, BOSS, PROP, TSBF and DTW without multi-scale convolution and attention mechanism, and better than inception time and OS-CNN using multi-scale convolution.

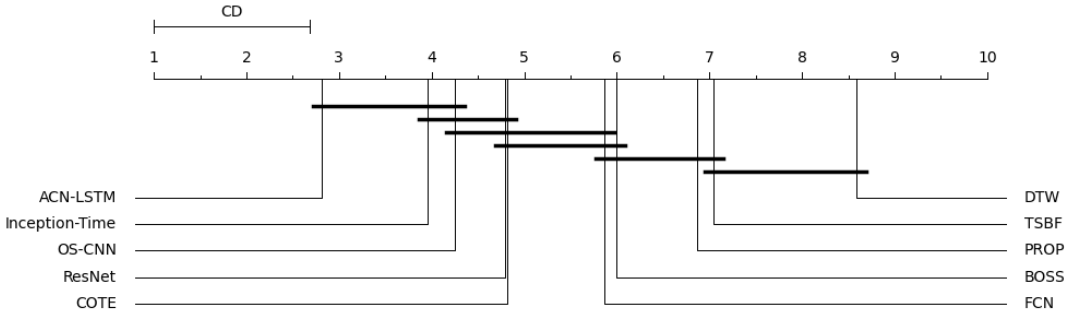

Fig. 5 Critical difference diagram based on the average arithmetic ranks

Fig.6 shows the paired comparison scatter of all models and ACN-LSTM. Each point denotes the accuracy of two classifiers on a dataset. The farther the point is from the diagonal, the greater the difference in accuracy. The point above the diagonal indicates that the proposed model is more accurate than the comparison method, and vice versa. Fig.6 shows that most of the points are above the diagonal, and there is a big gap between ACN-LSTM and the comparison method on many datasets, which proves that ACN-LSTM is superior to other

methods.

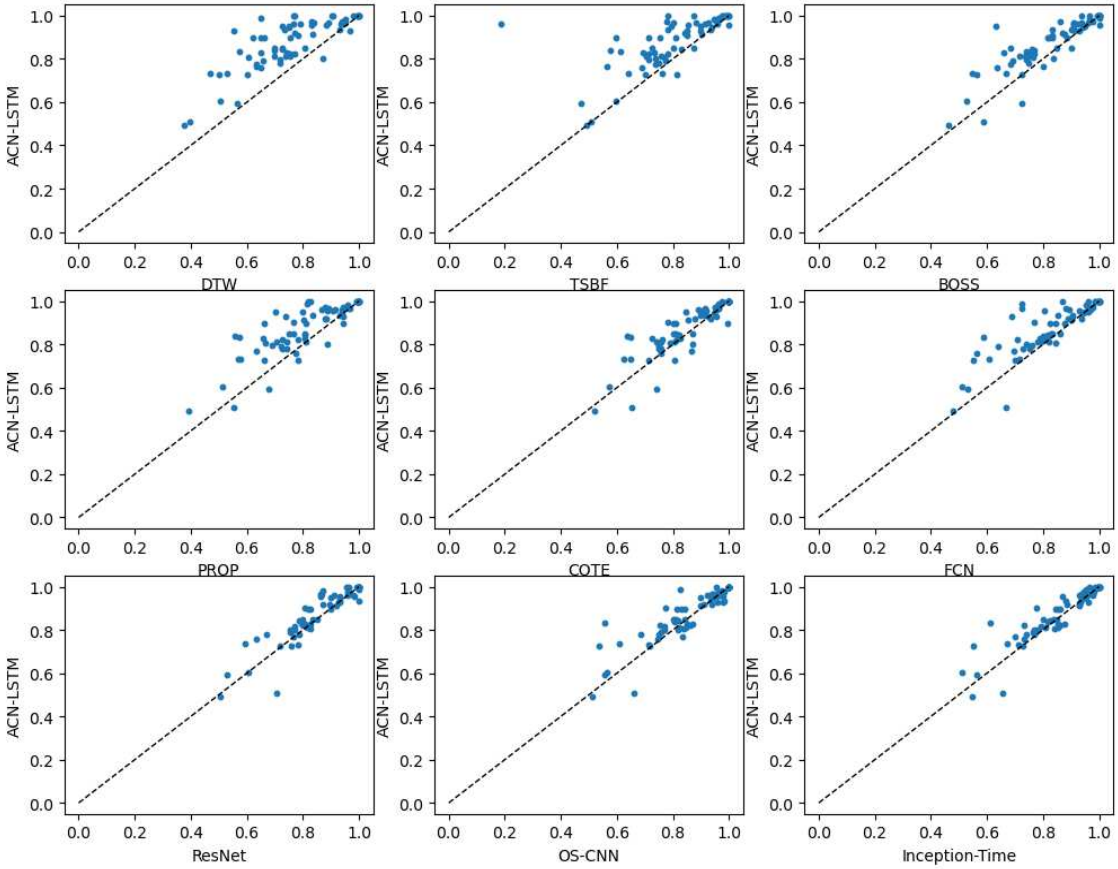

Fig. 6 . Scatter plots of pairwise comparison of all models against ACN-LSTM

Tab.5 shows the comparison of average top-1 accuracy on datasets with different sequence lengths. It is seen from Tab. 5 that COTE performs better than DTW, TSBF, boss and prop on short, medium and long sequence datasets, while OS-CNN and inception time perform better than COTE on short and small datasets, which proves the feasibility of applying the method based on deep learning to TSC tasks. ACN-LSTM achieves the highest accuracy when classifying on short and medium sequence datasets, while COTE method achieves the highest accuracy on long sequence datasets, but it is similar to the classification ability of ACN-LSTM model. When the sequence length is long, features can only be extracted on a relatively short length, which leads to limited improvement in the classification accuracy of ACN-LSTM for long datasets. COTE is an integrated method based on multiple algorithms, which has more advantages for the analysis of complex problems, which shows that this model is more conducive to dealing with short and medium length datasets.

Tab. 5 Average Top-1 Accuracy of Different models grouped by Serial Length

Length	DTW	TSBF	BOSS	PROP	COTE	FCN	ResNet	OS-CNN	Inception-Time	ACN-LSTM
<200	0.782	0.836	0.834	0.821	0.864	0.863	0.883	0.876	0.872	**0.895**
200-500	0.752	0.804	0.839	0.808	0.857	0.816	0.859	0.865	0.868	**0.886**
>500	0.698	0.698	0.795	0.768	**0.814**	0.767	0.785	0.794	0.802	0.813

Tab. 6 shows the comparison of average top-1 accuracy on datasets in different fields. It is seen from Tab. 6 that ACN-LSTM achieves the best performance in different domains except for device and motion datasets.

Almost all datasets of the device category are long sequences. Compared with data in other fields, all existing methods have higher error rates on these datasets, which means that TSC tasks in the device domain are more complex. COTE has the highest accuracy, indicating that COTE is more suitable for such complex classification tasks, and ACN-LSTM is more suitable for short and medium length sequences.

Tab. 6 Average Top-1 Accuracy of Different models grouped by domain

Domain	DTW	TSBF	BOSS	PROP	COTE	FCN	ResNet	OS-CNN	Inception-Time	ACN-LSTM
Device	0.566	0.610	0.698	0.650	**0.712**	0.681	0.692	0.690	0.689	0.660
ECG	0.763	0.885	0.937	0.879	0.941	0.939	0.919	0.950	0.955	**0.957**
Image	0.720	0.751	0.796	0.756	0.797	0.780	0.814	0.806	0.812	**0.850**
Motion	0.657	0.661	0.674	0.698	0.742	0.718	0.736	0.772	**0.778**	0.739
Sensor	0.768	0.802	0.847	0.838	0.892	0.876	0.887	0.882	0.885	**0.900**
Simulated	0.822	0.869	0.880	0.871	0.908	0.865	0.902	0.906	0.910	**0.936**
Spectro	0.758	0.815	0.842	0.784	0.857	0.817	0.867	0.883	0.837	**0.888**

Based on the above experimental results, ACN-LSTM has achieved better comprehensive performance than the comparison method on 65 experimental datasets, which shows that the combination of multi-scale convolution and attention mechanism can make the network achieve better classification performance. Therefore, compared with other models based on deep learning, ACN-LSTM has more advantages in solving univariate time series classification tasks.

4.3.2 Convergence comparison

For convergence evaluation, the dataset with obvious differences between two different models is select, and then the error rate of each epoch to observe the convergence process is recorded. Fig. 7 shows that for the dataset Arrow Head, compared with other models, the models ACN-LSTM and OS-CNN proposed in this paper have lower error rate and faster convergence speed, and ACN-LSTM can achieve a lower error rate at the beginning of training, and the convergence effect is better. Since there is no attention module in inception time, the error rate is relatively high, and the convergence is unstable. For FCN and RESNET models, the convergence is good, but the accuracy is relatively low, which may explain the high average error rate caused by the insufficient complexity of the model. For ECGFiveDays, the models ACN-LSTM and OS-CNN proposed in this paper have lower error rate and faster convergence speed.

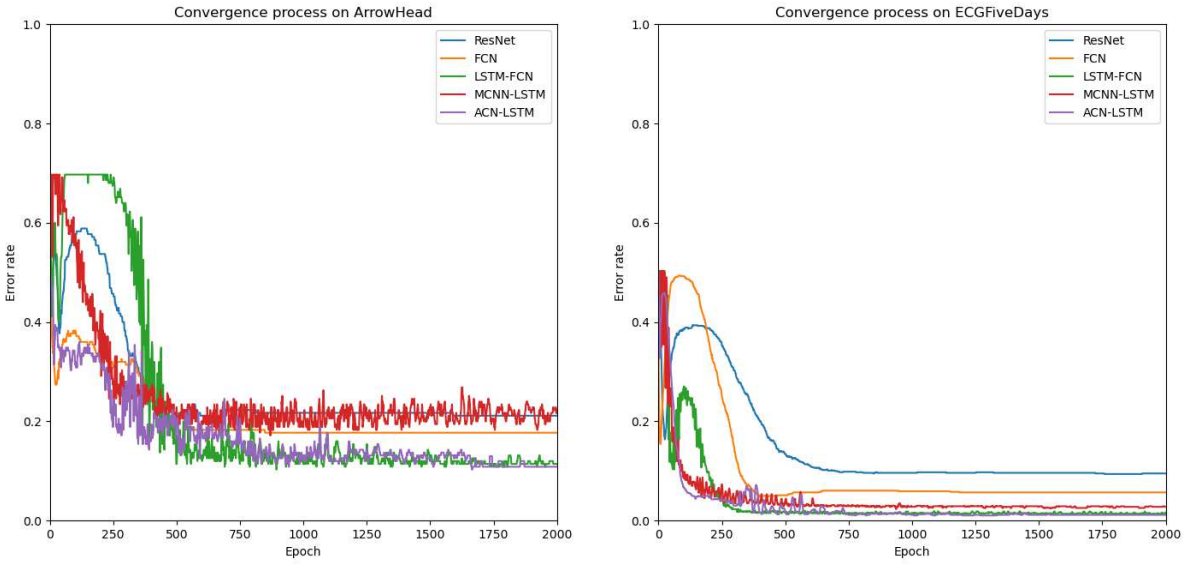

Fig. 7 Convergence process of different deep learning models

4.3.3 Performance of binary classification

Binary classification is a basic problem of TSC task, and 25 of the 65 data sets selected are binary classification tasks. Therefore, to observe the performance difference of each model in binary classification, the ROC curve of each model on ham data set is drawn, and AUC-ROC is used to evaluate the performance. As shown in Fig.8, ACN-LSTM achieves the maximum AUC-ROC value, which indicates that ACN-LSTM is the best deep learning model

for binary classification.

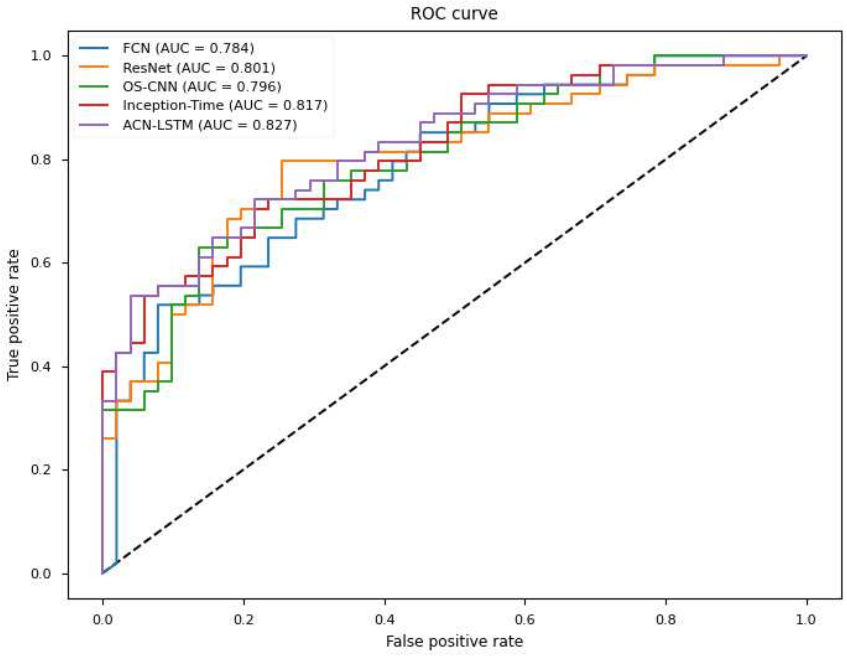

Fig. 8 ROC curves of different deep learning models.

4.4 Visualization

Class activation map (CAM) [41] was first applied in the field of image classification, aiming to find the region that contributes the most in the image. Later, Wang [42] and Fawaz [43] introduced one-dimensional CAM for visualization to find the subsequences of time series that contribute the most to classification.

Let $A_m(t)$ be the result of the last convolution t time, where $m \in [1, M], t \in [1, T]$. ω_m^c is the weight between the m filter and the output neuron of class c. Due to the use of the global average layer, the input of the cth class of neurons z_c is calculated as follows:

$$z_c = \sum_m \omega_m^c \sum_t A_m(t) = \sum_t \sum_m \omega_m^c A_m(t) \tag{26}$$

Finally, the result of time t is as follows:

$$CAM_c(t) = \sum_m \omega_m^c A_m(t) \tag{27}$$

Therefore, the value of $CAM_c(t)$ indicates the importance of activation at time t, and the timestamp t generates the classification result c. By simply sampling CAM up to the size of the original time series, the most important subsequences are identified.

CAM is applied to datasets Gun to understand the impact of multiscale convolution and attention mechanisms. Since shapelet found and interpreted the discriminant subsequence of gun dataset, which is to understand the visualization results and prove the effectiveness of ACN-LSTM.

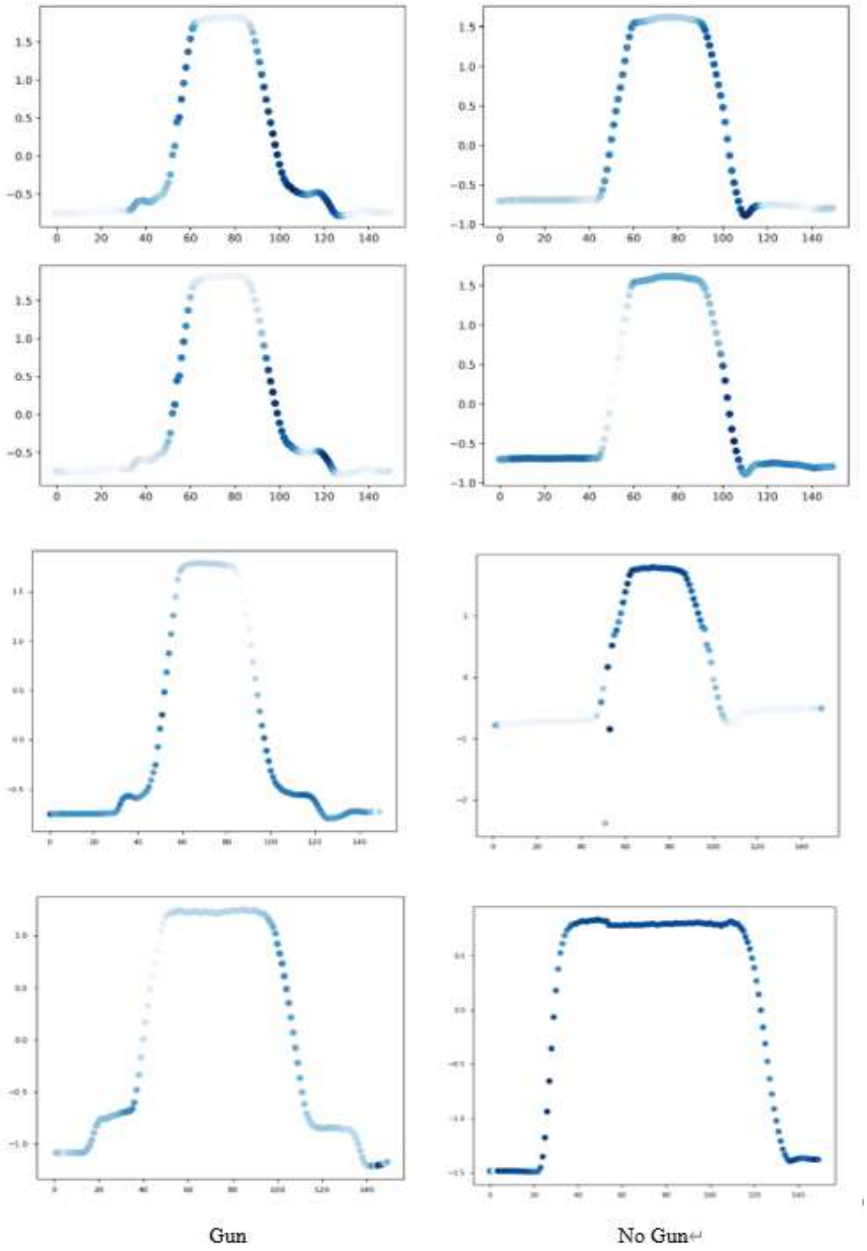

Fig. 9 CAM results of different models on the Gun Point dataset

Fig. 9 shows the CAM results of FCN, RESNET, OS-CNN and ACN-LSTM models on gun dataset in turn. It is found from the figure that the discriminative regions of all models are concentrated on the right side of the whole sequence, which shows that the right side of the time series plays a role in decision-making. The darker color length of FCN and RESNET

models is much shorter than that of ACN-LSTM, which indicates that FCN and RESNET cannot capture the characteristics of the whole region due to single-scale convolution, while multi-scale convolution generates receptive fields of different sizes to capture information of different scales, which solves the limitations of single-scale convolution. Since the right part of the time series plays a decisive role in distinguishing it, compared with the OS-CNN model without attention mechanism, it is seen from the CAM diagram of ACN-LSTM that the right part of the time series is the only discriminative area, eliminating the influence of redundant information, which shows that, due to the combination of multi-scale convolution and attention mechanism, the discrimination region is larger and more accurate. The above comprehensive comparison and evaluation shows that the TSC task method based on deep learning is feasible, and ACN-LSTM has specific and superior performance.

5. Conclusions and future work

To improve the classification accuracy of univariate time series, an LSTM network structure ACN-LSTM based on adaptive convolution and attention mechanism is proposed in this paper. The network is a parallel architecture composed of two branches, combined with LSTM, multi-scale convolution module and channel adaptive feature adjustment module. LSTM controls the transmission of sequence information, such that the network can effectively learn the time-related information. The multi-scale convolution module extracts the features of time series data with different scales by generating multiple sensing domains to obtain richer feature presentation. The channel adaptive feature adjustment module fuses the channel feature information to make the network focus on important features. The experiment and evaluation results on 65 UCR datasets show that ACN-LSTM has better performance than the other 9 methods. However, the classification effect of ACN-LSTM on long time series is not significantly improved. We have not yet applied ACN-LSTM to the multivariable time series classification dataset. In future work, we will further study the model architecture to solve the multivariable and long series time series classification problems based on ACN-LSTM.

Acknowledgements

This work was partially supported by the Natural Science Foundation of Hubei Province (No. 2020CFB546), National Natural Science Foundation of China under Grants 12001411, and the Fundamental Research Funds for the Central Universities (WUT: 2020IA004, 2020-IB-003).

References

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

[2] Z. Xing, J. Pei, and E. Keogh. A brief survey on sequence classification. ACM SIGKDD Explorations Newsletter, 12(1):40-48, 2010

[3] 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. 2017, 31(3):606–660

[4] 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.

[5] 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.

[6] 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).

[7] Rakthanmanon T, Keogh E. Fast shapelets: A scalable algorithm for discovering time series shapelets[C]//proceedings of the 2013 SIAM International Conference on Data Mining. Society for Industrial and Applied Mathematics, 2013: 668-676.

[8] J. Lin, Eamonn J. Keogh, Li Wei, Stefano Lonardi: Experiencing SAX: a novel symbolic presentation of time series. DMKD, 15, 2, 2007, 107-144.

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

[10] Bagnall A, Lines J, Hills J, et al. Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles[J]. IEEE Transactions on Knowledge and Data Engineering, 2015, 27(9):1-1.

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

[12] Zhu H , Zhang J , Cui H , et al. TCRAN: Multivariate time series classification using residual channel attention networks with time correction. 2021.

[13] K. Muhammad, A. Ullah, A.S. Imran, M. Sajjad, M.S. Kiran, G. Sannino, V.H.C. de Albuquerque, et al., Human action recognition using attention based LSTM network with dilated CNN features, Future Gener. Comput. Syst. 125 (2021) 820–830.

[14] S. Kwon, Optimal feature selection based speech emotion recognition using two-stream deep convolutional neural network, Int. J. Intell. Syst. (2021).

[15] SHELHAMER E, LONG J, DATTELL T. Fully Convolutional Networks for Semantic Segmentation [J]. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2017, 39(4): 640-651.

[16] Z. WANG, W. YAN, T. OATES. Time series classification from scratch with deep neural networks: A strong baseline[C]// Proceedings of the International Joint Conference on Neural Networks: USA: Alaska, 2017: 1578–1585.

[17] CUI Z, CHEN W, CHEN Y. Multi-Scale Convolutional Neural Networks for Time Series Classification [EB/OL]. (2016-05-11) [2021-06-20]. https://arxiv.org/abs/1603.06995v4.

[18] KARIM F, MAJUMDAR S, DARABI H. Insights into LSTM Fully Convolutional Networks for Time Series Classification [J]. IEEE Access, 2019, 7: 67718-67725.

[19] KARIM F, MAJUMDAR S. DARABI H, CHEN S. LSTM Fully Convolutional Networks for Time Series Classification [J]. IEEE Access, 2018, 6: 1662-1669.

[20] CHEN W, SHI K, Multi-scale Attention Convolutional Neural Network for time series classification [J]. Neural Networks, 2021, 136: 126-140.

[21] F. Karim, S. Majumdar, H. Darabi, S. Harford, Multivariate LSTM-FCNs for time series classification, Neural Netw. 116 (2019) 237–245.

[22] Z XIAO, X XU, H XING, et al. RTFN: A Robust Temporal Feature Network for Time Series Classification [EB/OL]. (2020-12-29) [2021-06-08]. https://arxiv.org/abs/2011.11829.

[23] FAWAZ H I, LUCAS B, FPRESTIER G, et al. Inception-time: finding alexnet for time series classification [J]. Data Mining and Knowledge Discovery, 2020, 34: 1936–1962.

[24] SZEGEDY C, LIU W, JIA Y, et al. Going Deeper with Convolutions [C]// Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. USA: Boston.2015:1-9.

[25] TANG.W, LONG.G, LIU. L, ZHOU. T, J. et al. Rethinking 1d-cnn for time series classification: a stronger baseline [EB/OL]. (2021-02-12) [2021-6-13]. https://arxiv.org/abs/2002.10061v1.

[26] S.H. Huang, L. Xu, C. Jiang. Residual attention net for superior cross-domain time sequence modeling. arXiv preprint arXiv: 2001.04077, 2020.

[27] HOCHREITER S, SCHMIDHUBER J, Long Short-Term Memory [J]. Neural computation, 1997, 9(8): 1735–1780.

[28] A. Graves et al. , Supervised Sequence Labelling with Recurrent Neural Networks. Springer, 2012, vol. 385.

[29] CARRASCO M, BARBOT A. Spatial attention alters visual appearance [J]. Current Opinion in Psychology, 2019, 29: 56-64.

[30] LECUN Y, BSEER B, DENKER J S, et al. Backpropagation applied to handwritten zip code recognition [J]. Neural Computation, 1989, 1(4): 541-551.

[31] 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).

[32] JIE H, LI S, GANG S, Squeeze-and-Excitation Networks [C]// Proceedings of the Conference on Computer Vision and Pattern Recognition. USA: Salt Lake City, 2018: 7132-7141.

[33] Li X, Wang W, Hu X , et al. Selective Kernel Networks[J]. IEEE, 2020.

[34] 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).

[35] SRIVASTAVA N, HINTON G, KRIZHEVSKY A, et al. Dropout: A Simple Way to Prevent Neural Networks from Overfitting [J]. Machine Learning Research, 2014, 15(1): 1929-1958.

[36] IOFFE S, SZEGEDY C. Batch normalization: Accelerating deep network training by reducing internal covariate shift [C]// Proceedings of the 32nd International Conference on Machine Learning. France: Lille. 2015: 448–456.

[37] Nair V, G E HINTON. Rectified linear units improve restricted Boltzmann Machines [C]// Proceedings of the 27th International Conference on Machine Learning. Israel: Haifa. 2010: 807–814.

[38] F. Wang, M. Jiang, C. Qian, S. Yang, C. Li, H. Zhang, X. Wang, X. Tang, Residual attention network for image classification, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 3156–3164.

[39] KINGMA D P, BA J. Adam: A Method for Stochastic Optimization [EB/OL]. (2017-01-30) [2021-05-29]. https://arxiv.org/abs/1412.6980.

[40] CHOLLET F, et al. Keras [OL]. URL: https://github.com/ fchollet/keras, 2015.

[41] 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).

[42] 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).

[43] 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.