Lei and Wu *EURASIP Journal on Wireless Communications and Networking* (2020) 2020:46
https://doi.org/10.1186/s13638-020-1661-4

EURASIP Journal on Wireless Communications and Networking

RESEARCH Open Access

Time series classification based on statistical features

Yuxia Lei* and Zhongqiang Wu

Abstract

This paper presents a statistical feature approach in fully convolutional time series classification (TSC), which is aimed at improving the accuracy and efficiency of TSC. This method is based on fully convolutional neural networks (FCN), and there are the following two properties: statistical features in data preprocessing and fine-tuning strategies in network training. The key steps are described as follows: firstly, by the window slicing principle, dividing the original time series into multiple equal-length subsequences; secondly, by extracting statistical features on each subsequence, in order to form a new sequence as the input of the neural network, and training neural network by the fine-tuning idea; thirdly, by evaluating the classification performance about test sets; and finally, by comparing the sample sequence complexity and network classification loss accuracy with the FCN using the original sequence. Our experimental results show that the proposed method improved the classification effects of FCN and the residual network (ResNet), which means that it has a generalization ability to the network structures.

Keywords: Time series classification, Statistical features, Full convolutional neural network

1 Introduction

Time series data are widely used for representing special data such as biological observations, stock prices, weather readings, and health monitoring data. How to extract useful information from time series data is becoming more and more important. Time series classification (TSC) is one of the key tasks, which is aimed at predicting class labels for time series. Nowadays, TSC is widely used successfully in medical diagnosis and early warning [1].

Many scholars have studied TSC and provided some methods. The approaches can be divided into the following three categories: distance-based approach, feature-based approach, and ensemble-based ones [2]. The key idea of distance-based approach is to measure the similarity between any given two time series. Commonly used classification algorithms are *K*-nearest neighbor (KNN), support vector machines (SVM) with similarity-based kernels, and dynamic time warping (DTW). The key idea of feature-based ones is mainly extracting a set of features representing the global time series patterns, and then commonly form a bag-of-words (BoW) provided to classifiers [3]. These kind of classification algorithms are the bag-of-features framework (TSBF) [4], bag-of-SFA-symbols (BOSS) [5], and BOSSVS method [6]. The key idea of third ones is to combine different classifiers together to achieve a higher accuracy. For example, PROP combines 11 classifiers based on elastic distance measures with a weighted ensemble scheme [7]. Shapelet ensemble (SE) provides the classifiers through the shapelet transform in conjunction with a heterogeneous ensemble [8]. In summary, these methods described above commonly require data preprocessing and feature extraction.

In recent years, deep neural networks have been used for TSC tasks, for example, multi-scale convolutional neural network (MCNN) [1], fully convolutional network (FCN) [2], and residual network (ResNet) [2]. FCN and ResNet do not require a lot of data preprocessing or feature engineering and can provide a good classification effect. Karim et al. proposed long-short-term memory fully convolutional networks (LSTM-FCN), which further improved the performance of the FCN [9]. But it increased the complexity of calculations and classification. Based on FCN, this paper studies the time series preprocessing and found a way to improve the performance of FCN classification. The key idea is described as follows:

* Correspondence: Yx_lei@126.com; 17862328070@163.com
School of Information Science and Engineering, Qufu Normal University, Rizhao 276826, China

given some original time series of UCR data sets [10], the method uses the window slicing principle to divide the original time series into multiple equal-length subsequences and then extracts statistical features on each subsequence to form a new sequence as the input of the neural network. In the network training process, we introduce the fine-tuning idea and evaluate the classification performance of the network by using a test set. Finally, the sample sequence complexity and network classification loss accuracy are compared with the FCN using the original sequence. Experimental results show that FCN performs better on the samples of the statistical features of the extracted subsequences, and the found method also improves the classification performance of ResNet, which means that it has generalization ability to the network structures.

The remainder of the paper is organized as follows. Section 2 gives some basic notations such as time series classification. Section 3 introduces the network architecture and fine-tuning of models. Section 4 provides experiments and results analysis. Section 5 improves the experiments from two aspects: increase the number of original sequences and extract more statistical features. Finally, we present conclusions and future work.

2 Some basic notations

This section will introduce some basic notations such as TSC [11].

2.1 Definition 1

An ordered set of real values with the form $X = [x_1, x_2, ..., x_t]$ is called a univariate time series (UTS). The length of X is equal to number t.

2.2 Definition 2

Given N different UTS with $X^i \in R^T$, $X = [X^1, X^2, ..., X^N]$ is called an M-dimensional multivariate time series (MTS).

2.3 Definition 3

A data set $D = \{(X_1, Y_1), (X_2, Y_2), ..., (X_N, Y_N)\}$ is a set of pairs (X_i, Y_i) where X_i could either be a UTS or MTS with Y_i as its one-hot label vector. For a data set containing N classes, Y_i is the vector of length N, where $j \in [1, N]$ is equal to 1 ifX_i is j and 0 otherwise.

2.4 Definition 4

Given some time series of length t, $X = \{X^t, t \in 1, 2, ...N\}$ and their labels $\{Y_1, Y_2, ..., Y_N\}$, TSC refers to input a new time series and output its labels Y_i.

3 Network architecture and fine-tuning of models

3.1 Network module and architecture

In FCN, a time convolution block is used as a time series feature extraction module, which consists of a convolution layer, a batch normalization layer, and an activation function [4, 12–15]. Some researchers provided methods for training deep networks [15–18]. The activation function may be a linear correction unit or a parametric linear correction unit. Convolution layer is shown in Fig. 1. Generally, the input to a temporal convolutional network is a time series signal. Given L convolutional layers, filters can capture how the input signals evolve over the course of an action. For each layer, all of the filters are parameterized by tensor $W^{(l)} \in R^{F_l \times d \times F_{l-1}}$ and biases $b^{(l)} \in R^{F_l}$, where $l \in \{1, ..., L\}$ is the layer index and d is the filter duration. For the l – th layer, the i – th component of the incoming (normalized) activation matrix $E^{(l-1)} \in R^{F_{l-1} \times T_{l-1}}$ from the previous layer for each time t where $f(\cdot)$ is a rectified linear unit [9, 12, 15].

$$\widehat{E_{i,t}^{(l)}} = f\left(b_i^{(l)} + \sum_{t'=1}^{d} \left\langle W_{i,t',.}^{(l)}, E_{.,t+d-t'}^{(l-1)} \right\rangle\right) \tag{1}$$

When training neural networks, standardized input can increase the speed of neural network training. The method is to normalize the training data, which means that raw data is subtracted from its mean and divided by its variance. Standard processing is also required at each hidden layer of the neural network, which is the batch normalization, as shown in Fig. 2.

Batch normalization processes lth hidden layer input Z^{l-1} as follows, ignoring the superscript $l - 1$:

$$\begin{aligned} \mu &= \frac{1}{m}\sum_i z^{(i)} \\ \sigma^2 &= \frac{1}{m}\sum_i (z_i - \mu)^2 \\ z_{\text{norm}}^{(i)} &= \frac{z^{(i)} - \mu}{\sqrt{\sigma^2 + \varepsilon}} \end{aligned} \tag{2}$$

where m is the number of single mini-batch samples and ε prevents the denominator from being zero, which can

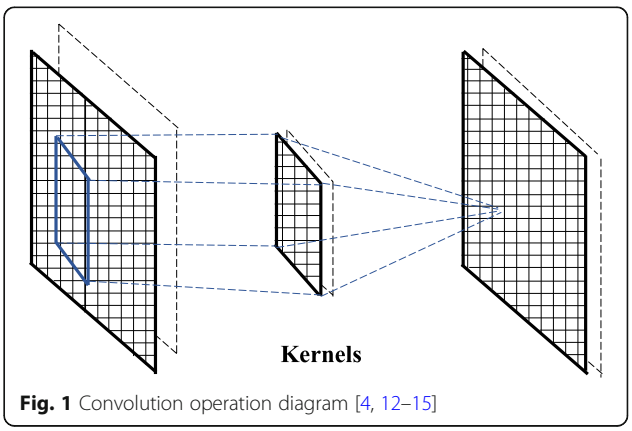

Fig. 1 Convolution operation diagram [4, 12–15]

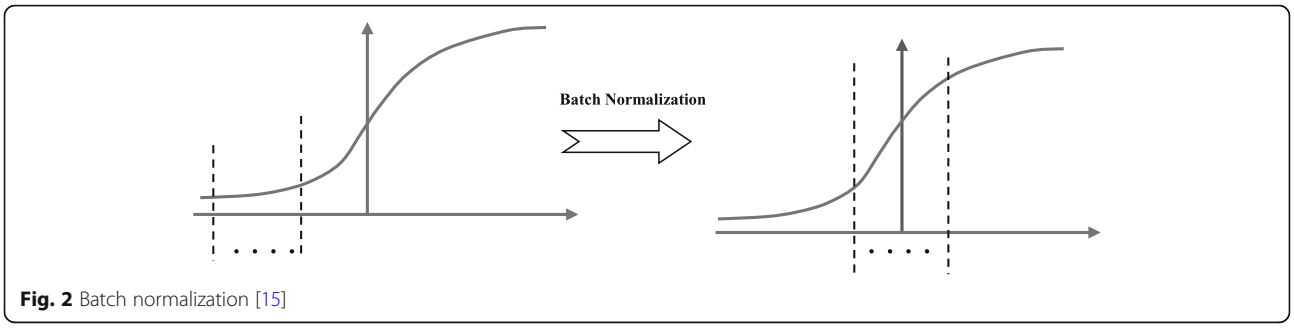

Fig. 2 Batch normalization [15]

be a value 10^{-8}. In this way, all the inputs of the hidden layer have a mean of 0 and a variance of 1.

In most cases, we do not expect all $z^{(i)}$ means equal to 0 and the variance is 1. So, it usually needs to be further processed on$z^{(i)}$:

$$\tilde{z^{(i)}} = \gamma \cdot z^{(i)}_{\text{norm}} + \beta \tag{3}$$

where γ and β are hyper-parameters, which can be obtained by gradient descent. Setting γ and β to different values, you can get any mean and variance. Activation function ReLU is shown in Fig. 3. Compared with other activation functions, ReLU has the advantages as follows: for linear functions, ReLU is more expressive; for nonlinear functions, ReLU is due to nonnegative interval gradients. Hence, there is no vanishing gradient problem.

After the time series is extracted by the time convolutional block, the extracted features are input to the global average pooling module to output the classification result. The global average pooling module is composed of the global average pooling layer and the Softmax layer. Global average pooling layer is shown in Fig. 4.

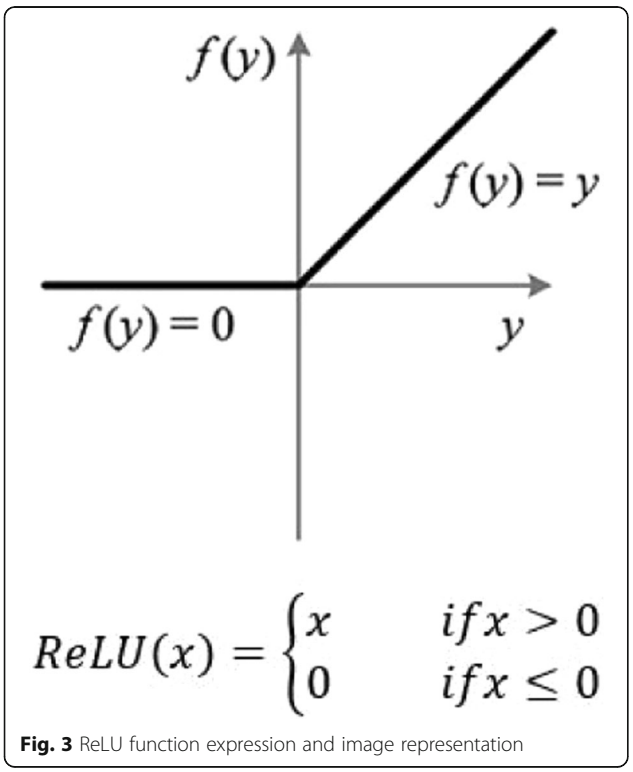

Fig. 3 ReLU function expression and image representation

The last layer of the traditional CNN is a fully connected layer. The number of parameters is very large, which is easy to cause over-fitting (such as AlexNet network). Therefore, the paper proposes to use the global mean pooling layer instead of the fully connected layer [13]. For different traditional connection layers, global averaging is performed for each feature sequence so that each feature sequence can get an output. This can greatly reduce the network parameters and avoid overfitting. On the other hand, each feature sequence is equivalent to an output feature, and this feature represents the characteristics of the output class.

TSC is multi-category. Generally, the most common method for solving multi-classification is to design n-output nodes. For example, the neural network can design an n-dimensional array as an output. Each dimension in the array corresponds to a category of the time series. The Softmax layer transforms the output of a neural network into a probability distribution. Each dimension value represents the probability of belonging to this category, and hence by cross entropy, it can be predicted about the distance between the probability distribution and one of the real answers.

FCN is performed as a feature extract. Its overall structure is shown in Fig. 5. The FCN contains three time convolutional blocks and one global average pooling one, where filter sizes of convolutional blocks are 128, 256, and 128, respectively.

The convolution operation is fulfilled by three 1D kernels without striding, whose sizes are 8, 5, and 3, respectively. The 1 × 1 convolution kernel can operate on a cross-channel aggregation and further reduce the dimension. The calculation of the time convolution block [2] is as follows:

$$\begin{aligned} y &= W \otimes x + b \\ s &= \text{BN}(y) \\ h &= \text{ReLU}(s) \end{aligned} \tag{4}$$

where $\otimes$ is the convolution operator.

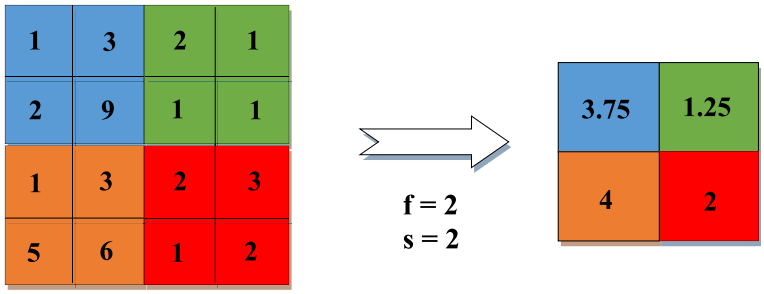

Fig. 4 Schematic diagram of pooling operation

The global average pooling block further processes the feature sequence, and the final classification result is given by the Softmax layer. Residual network is shown in Fig. 6. By adding a long jump connection to each fragment, ResNet extends the neural networks to very deep structures, which can allow the gradient flow to pass directly through the bottom layer. It achieves the most advanced performance in target detection and other visually related tasks [14]. The residual network also performs well in solving TCS. The convolutional blocks in Eq. 4 are used to build each residual block. Let Block_k denote the convolutional block with the number of filters k, and the residual block is formalized as follows [2]:

$$\begin{aligned} h_1 &= \text{Block}_{k_1}(x) \\ h_2 &= \text{Block}_{k_2}(h_1) \\ h_3 &= \text{Block}_{k_3}(h_2) \\ y &= h_3 + x \\ \hat{h} &= \text{ReLU}(y) \end{aligned} \tag{5}$$

The number of filters is the pair (64, 128, 128). The residual network used consists of three residual blocks and a global mean pool block, as shown in Fig. 6.

3.2 Fine-tuning of models

Fine-tuning can be described as transfer learning on the same data set [9]. The training procedure consists of two distinct phases. In the initial phase, the model is trained on a given data set with selected optimal hyperparameters. In the second one, fine-tuning is applied to the initial model. In fact, the procedure of transfer learning is iterated over in fine-tuning phase. Each repetition is initialized using the model weight of the previous iteration. At each iteration, the learning rate is halved, which accelerates the iterative process. The procedure is repeated K times, until the initial learning rate gets the lowest threshold, where K is an arbitrary constant. The algorithm is shown in Algorithm 1.

Algorithm 1 Fine-tuning

for i < K do

$model_{weights} \leftarrow initial\_model_{weights}$

$Train(model, initial\_lr, batchsize)$

$inital\_model_{weights} \leftarrow model_{weights}$

$i \leftarrow i+1$

$inial\_lr \leftarrow updateLearningRate(inital\_lr, i)$

4 Experiments and result analysis

4.1 Experiment settings

In the experiment, FCN neural network is trained with original sequence data set and data set based on statistical feature processing, and the network performance is evaluated on the test set. In order to present the generalization ability of the neural network structure based on statistical feature sequence, the same experiment on ResNet is carried out to evaluate and compare. The original sequence data set comes from UCR time series data, which contains 44 classes, divided into training sets and test sets, as shown in https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. Among them, the class with the longer time series is selected as the

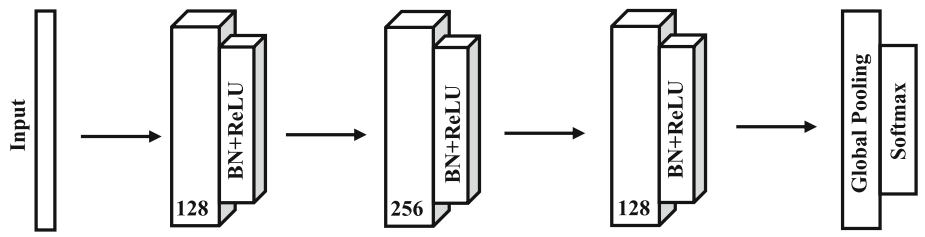

Fig. 5 The network structure of FCN [2]

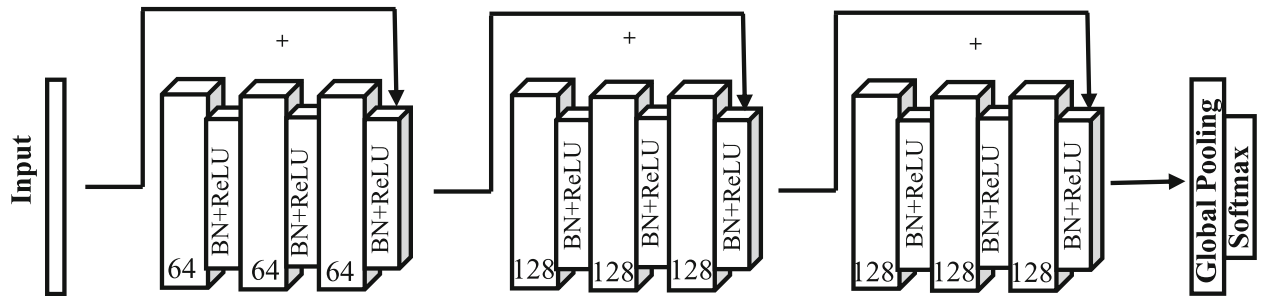

Fig. 6 The network structure of ResNet [2]

input sample. The training sample format in the data set is as follows:

$$\text{data} = \begin{bmatrix} y_1, x_1^1, x_2^1, \ldots, x_t^1 \\ y_2, x_1^2, x_2^2, \ldots, x_t^2 \\ \ldots\ldots \\ y_N, x_1^N, x_2^N, \ldots, x_t^N \end{bmatrix}_{N\times(t+1)} \tag{6}$$

where N is the number of training samples and t is the length of the time series in the training samples. The only preprocessing of the original sequence before the experiment is standardization. Let μ be the average and σ^2 be the variance of the data, then

$$\text{data} = \frac{\text{data}-\mu}{\sigma^2} \tag{7}$$

Then the training samples are extracted from the data to form a matrix.

$$X = \begin{bmatrix} X^1 \\ X^2 \\ X^3 \\ X^4 \end{bmatrix} = \begin{bmatrix} x_1^1, x_2^1, \ldots, x_t^1 \\ x_1^2, x_2^2, \ldots, x_t^2 \\ \ldots\ldots \\ x_1^N, x_2^N, \ldots, x_t^N \end{bmatrix}_{N\times t} \tag{8}$$

The class label y_i corresponding to the training sample is converted to a one-hot vector Y_i, which is composed of only 0 and 1. If the training sample X^i belongs to a class, the corresponding class label is 1, and otherwise is 0.

$$Y = [Y_1, Y_2, \ldots, Y_N] = \begin{bmatrix} 1,0,0,\ldots,0 \\ 0,1,0,\ldots,0 \\ 0,0,1,\ldots,0 \\ \ldots \\ 0,0,0,\ldots,1 \end{bmatrix}_{C\times N} \tag{9}$$

where C is the number of types of time series. Normalize the time series samples (X, Y) into the FCN neural network for training. We call statistical feature data set, which is shown in Fig. 7.

With regard to time series preprocessing, the traditional machine learning method is to extract the

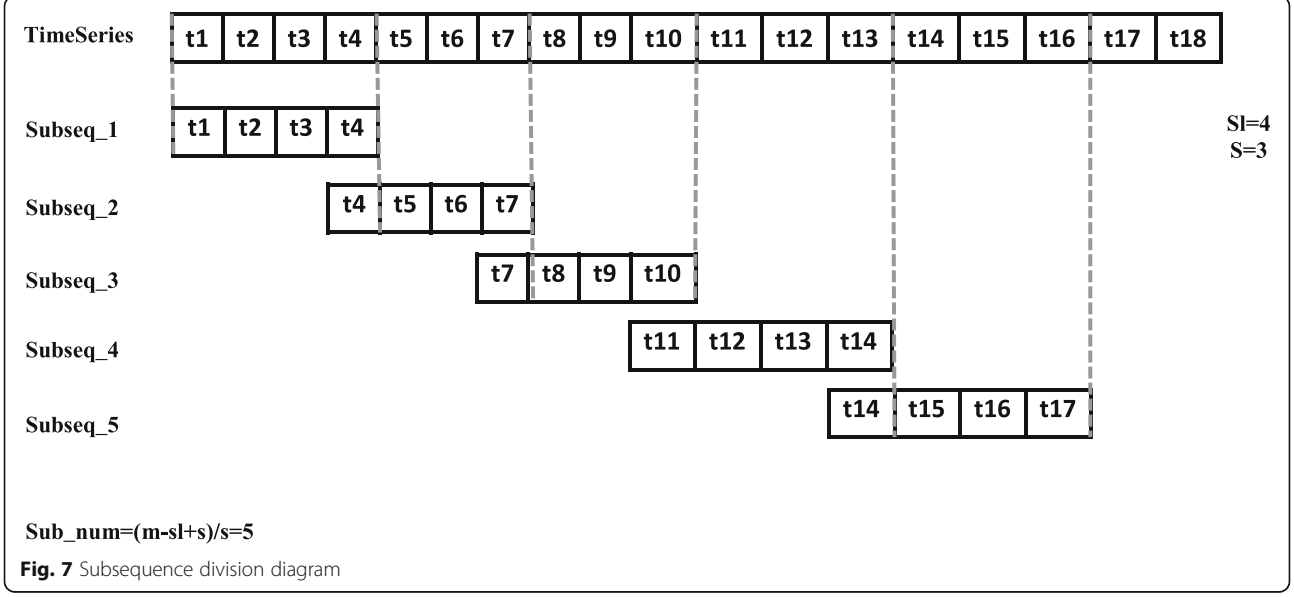

Fig. 7 Subsequence division diagram

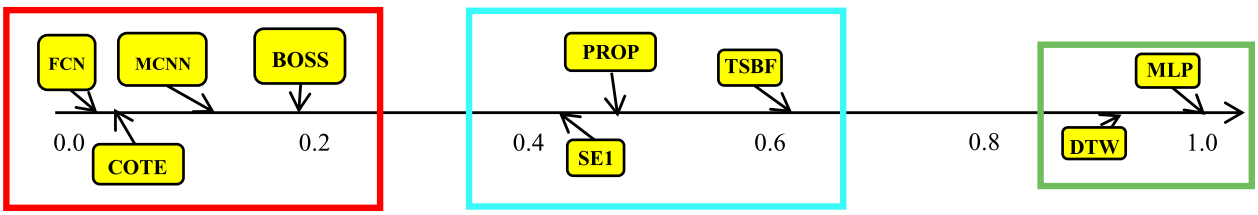

Fig. 8 MPCE ranking of time series algorithm

statistical features of the entire time series and then use the algorithm of random forest to classify. The modern deep learning method is to convolute directly on the normalized original sequence [2], then enter into FCN. In this paper, we have found a better time series preprocessing method. Before the standardization, the original sequence is divided into multiple subsequences according to the given step size and subsequence length, and then the statistical features are extracted from each subsequence (maximum, mean, polar difference, variance, peak trough, kurtosis, etc.), and let these statistical features constitute a new time series as the input of the neural network. On the data set, the final input data samples are described as follows:

$$\begin{bmatrix} x_1^1, x_2^1, \ldots, x_t^1 \\ x_1^2, x_2^2, \ldots, x_t^2 \\ \cdots \\ x_1^N, x_2^N, \ldots, x_t^N \end{bmatrix}_{N\times t} \underset{sl,s,nsf}{} \begin{bmatrix} x_1^1, \ldots, x_{\mathrm{nsf}}^1 & x_1^1, \ldots, x_{\mathrm{nsf}}^1 & \cdots \\ x_1^2, \ldots, x_{\mathrm{nsf}}^2 & x_1^2, \ldots, x_{\mathrm{nsf}}^2 & \cdots \\ \cdots & \cdots & \cdots \\ x_1^N, \ldots, x_{\mathrm{nsf}}^N & x_1^N, \ldots, x_{\mathrm{nsf}}^N & \cdots \end{bmatrix}_{N\times \mathrm{nsf}\frac{t-sl+s}{s}} = X_{\mathrm{new}} \tag{10}$$

where sl is the length of the subsequence, s denotes to stride, and nsf refers to the number of statistical features selected. Moreover, new time series samples (X_{new}, Y) are standardized and then input into FCN neural network for training.

4.2 Hyper-parameter setting

FCN and ResNet are trained with the Adam algorithm. Using the learning rate attenuation (the learning rate decreases as the number of iterations increases), the initial value of the learning rate is set to 0.001, and the final value of the fine-tuning 0.0001; in order to avoid the small amount of data and when the neural work is overfitting, the method of early termination is adopted, in which the optimal value does not change for 50 generations, and the network determines that the optimal value is found, and the program ends early. Batch normalization parameters $\beta_1 = 0.9$, $\beta_2 = 0.999$, $\varepsilon = 10^{-8}$. The loss function for all tested models is categorical cross entropy. The number of iterations is 2000.

4.3 Experiment and evaluation

In order to present the generalization ability based on the statistical feature method, the same experiments are performed on the residual network. The performance of FCN and ResNet has been presented to be superior to

Table 1 The dimensions of OTS and new SEF

Class	Original data		Subsequence data	
	Train_sample	Test_sample	Train_sample	Test sample
Beef	(30,470,1,1)	(30,470,1,1)	(30,17,7,1)	(30,17,7,1)
_ECG_torso	(1380,1639,1,1)	(40,1639,1,)	(1380,64,7,)	(40,64,7,1)
FISH	(175,463,1,1)	(175,463,1,1)	(175,17,7,1)	(175,17,7,1)
Haptics	(308,1092,1,1)	(155,1092,1,1)	(308,42,7,1)	(155,42,7,1)
nlineSkate	(550,1882,1, 1)	(100,1882,1,1)	(550,74,7,1)	(100,74,7,1)
Lighting2	(61,637,1,1)	(60,637,1,1)	(61,24,7,1)	(60,24,7,1)
MALLAT	(2345,1024,1,1)	(55,1024,1,1)	(2345,750,1,1)	(55,750,1,1)
CG_Thorax1	(1965,750,1, 1)	(1800,750,1,1)	(1965,29,7,1)	(1800,29,7,1)
CG_Thorax2	(1965,750,1, 1)	(1800,750,1,1)	(1965,29,7,1)	(1800,29,7,1)
OliveOil	(30,570,1,1)	(30,570,1,1)	(30,21,7,1)	(30,21,7,1)
OSULeaf	(242,427,1,1)	(200,427,1,1)	(242,6,7,1)	(200,16,7,1)
ightCurves	(8236,1024,1,1)	(1000,1024,1,1)	(8236,39,7,1)	(1000,39,7,1)
Symbols	(995,398,1,1)	(25,398,1,1)	(995,14,7,1)	(25,14,7,1)
yoga	(3000,426,1,1)	(300,426,1,1)	(3000,16,7,1)	(300,16,7,1)

Table 2 Comparison of loss, accuracy, and time of OTS and characteristic subsequences on FCN

Class	Sub_acc	Ori_acc	Sub_loss	Ori_loss	Sub_time (ms)	Ori_time (ms)	MPCE	
							Sub	Ori
Beef	0.20000000	0.20000000	1.74064850	1.78573286	3084	1027	0.0266	0.0266
_ECG_torso	*0.25000000*	0.12500000	0.75720222	0.95993692	2001	3002	0.0187	0.0219
FISH	0.12000000	0.16000000	1.89679190	2.04517563	1007	2009	0.0050	0.0048
Haptics	*0.25161290*	0.21935483	1.60219245	1.62430625	2007	2008	0.0048	0.0050
nlineSkate	0.14000000	0.21000000	1.87575633	1.92259577	2004	3006	0.0086	0.0079
Lighting2	*0.66666665*	0.58333333	0.7384292	0.66227742	2041	3046	0.0055	0.0069
MALLAT	*0.30909091*	0.12727272	1.18807646	1.60943770	4002	5002	0.0126	0.0159
CG_Thorax1	*0.06666666*	0.05833333	3.53328775	3.68092299	4002	5003	0.0005	0.0005
CG_Thorax2	*0.08777777*	0.04166666	3.56809761	3.69119312	4002	5003	0.0005	0.0005
OliveOil	0.16666667	0.26666668	1.39539515	1.85584843	4137	4149	0.0278	0.0244
OSULeaf	*0.30000000*	0.24500000	1.56651332	1.55764289	5020	55021	0.0035	0.0038
ightCurves	*0.91000000*	0.46100000	0.17210860	0.36349089	8942	11001	0.0001	0.0005
Symbols	*0.92000001*	0.36000001	0.75745789	1.02618091	6006	6006	0.0032	0.0256
yoga	0.45666666	0.49666666	0.58991410	0.67094716	7002	8003	0.0018	0.0017
mean	*0.346082*	0.253878	*1.527277*	1.675406	*3946.929*	8091.857	*0.0085*	0.0104

Data in italic means better experimental results

many other deep learning algorithms in dealing with the task of time series classification. According to the standardized MPCE score based on the mean *t* test given in [2], the rankings of FCN and ResNet are shown in Fig. 8. In the experiment, FCN and ResNet are trained with standardized raw data sets and new data sets based on statistical features, and then the test set is used for error and loss analysis to draw a conclusion. The experimental environment uses Google Colaboratory and the GPU model is Tesla K80. Google Colaboratory is a research tool open to Google, mainly used for machine learning development and research, where you can run python code to train neural networks.

Mean per class error (MPCE) is the arithmetic mean of each type of error, which can be used to evaluate the classification performance of the specific models on multiple data sets. The calculation formula is as follows:

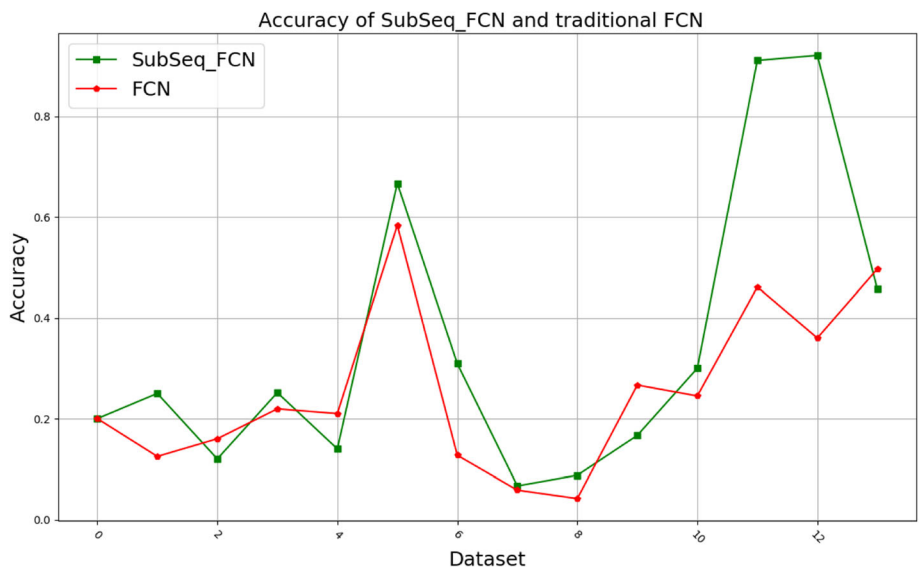

Fig. 9 The accuracy of the original sequence and feature subsequence on FCN

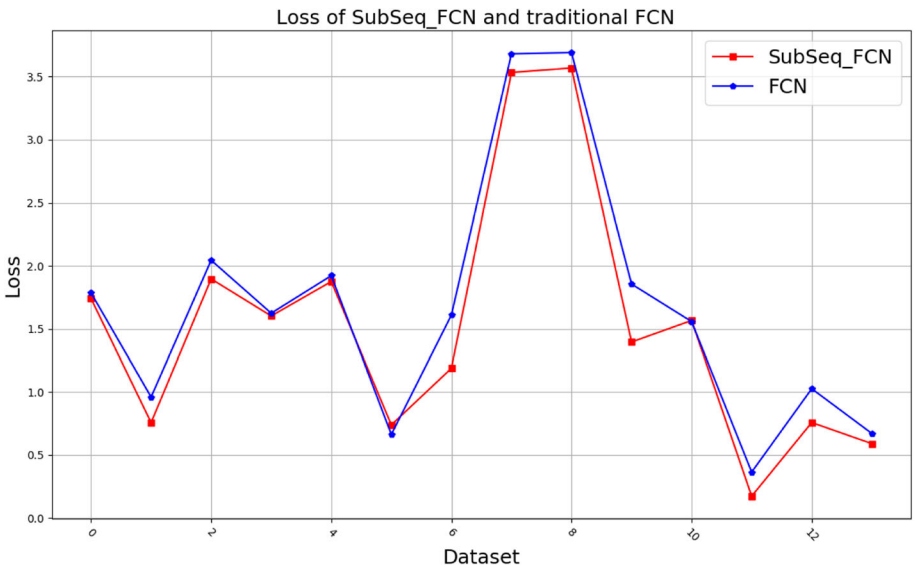

Fig. 10 The loss of original sequence and feature subsequences on FCN

$$\begin{aligned} \mathrm{PCE}_k &= \frac{1-\mathrm{acc}}{C_k} \\ \mathrm{MPCE}_i &= \frac{1}{K}\sum \mathrm{PCE}_k \end{aligned} \tag{11}$$

The basic idea of MPCE is as follows: the expected error rate for a single class across all the data sets. By considering the number of classes, MPCE is more robust as a baseline criterion [2].

4.4 Results and analysis

The experiment limited the length of the time series, selected 14 classes with a time series length exceeding 350, and calculated the original time series and the time series dimensions after extracting the statistical features. The dimensions of the two time series are shown in Table 1, where the original time series (OTS) and new sequence of extracted features (SEF).

From the data in Table 1, it can be found that the original time series is divided into multiple subsequences

Table 3 Comparison of loss, accuracy, and time of OTS and characteristic subsequences on ResNet

Class	Sub_acc	Ori_acc	Sub_loss	Ori_loss	Sub_time (ms)	Ori_time (ms)	MPCE	
							Sub	Ori
Beef	0.20000000	0.20000000	1.74496531	1.73106301	5158	3086	0.0266	0.0266
_ECG_torso	0.25000000	0.57500000	0.39711393	0.76636980	8006	12008	0.0187	0.0106
FISH	*0.29142857*	0.12000000	1.74508711	2.00773776	5028	6034	0.0040	0.0050
Haptics	*0.21935484*	0.11612903	1.59135229	1.63156084	80255	9029	0.0050	0.0057
nlineSkate	0.18000000	0.18000000	1.80085803	1.99357187	13023	13023	0.0082	0.0082
Lighting2	0.66666666	0.66666665	0.82555122	0.80486704	10167	11185	0.0055	0.0055
MALLAT	*0.81818181*	0.25454545	0.45194966	0.68730921	16007	22009	0.0033	0.0135
CG_Thorax1	*0.09777778*	0.08388889	3.28499700	3.29407895	18009	23012	0.0005	0.0005
CG_Thorax2	*0.13000000*	0.02277778	3.31574941	3.43187079	20010	25013	0.0005	0.0005
OliveOil	0.16666667	0.43333334	1.71029079	1.31717467	18609	20650	0.0278	0.0189
OSULeaf	*0.27000000*	0.22000000	1.52914039	1.45554120	21089	22093	0.0036	0.0039
ightCurves	*0.96600000*	0.84100000	0.11879947	0.24178773	36004	555007	0.0000	0.0001
Symbols	*0.87999999*	0.40000000	0.43127703	0.52733892	27027	30030	0.0048	0.0240
yoga	0.45666666	0.45666666	0.52088028	0.64692467	32011	36012	0.0018	0.0018
mean	*0.399482*	0.326429	*1.390572*	1.466943	*22171.64*	56299.36	*0.0079*	0.0089

Data in italic means better experimental results

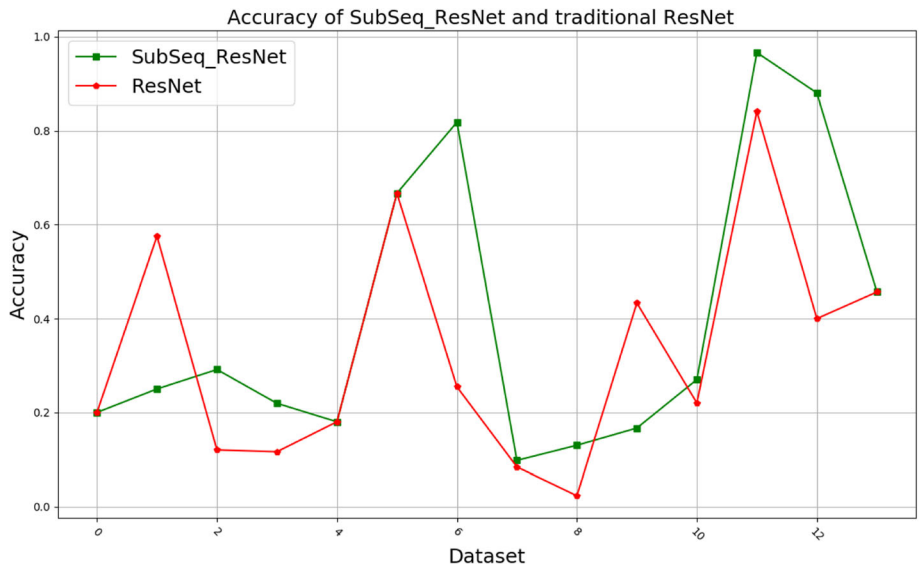

Fig. 11 The accuracy of the original sequence and feature subsequence on ResNet

and the data dimension of the new time series formed after extracting the statistical features is significantly reduced, and the new sequence maintains the statistical characteristics of the original series and removed redundant part, so if the new sequence is input to FCN, the calculation of the neural network can be simplified and the operation efficiency can be improved while ensuring the correct rate. The original sequence and the extracted features are first tested in FCN. The final loss and accuracy of the various types, as well as the final results of the network's runtime, are shown in Table 2.

From Table 2, in many time series categories, the new sequence based on statistical features is more accurate in the FCN than the original sequence in terms of the loss function, and the overall performance is better. The FCN network has a short classification time and a small average error. The loss and accuracy of the original sequence and the new sequence based on statistical features are obtained and visualized. Accuracy images of the original sequence and feature subsequence on the FCN network, as shown in Fig. 9, the results show that the accuracy of the FCN based on the statistical feature subsequence is superior to the original sequence in most of the classes (the

Fig. 12 The loss of the original sequence and feature subsequences on ResNet

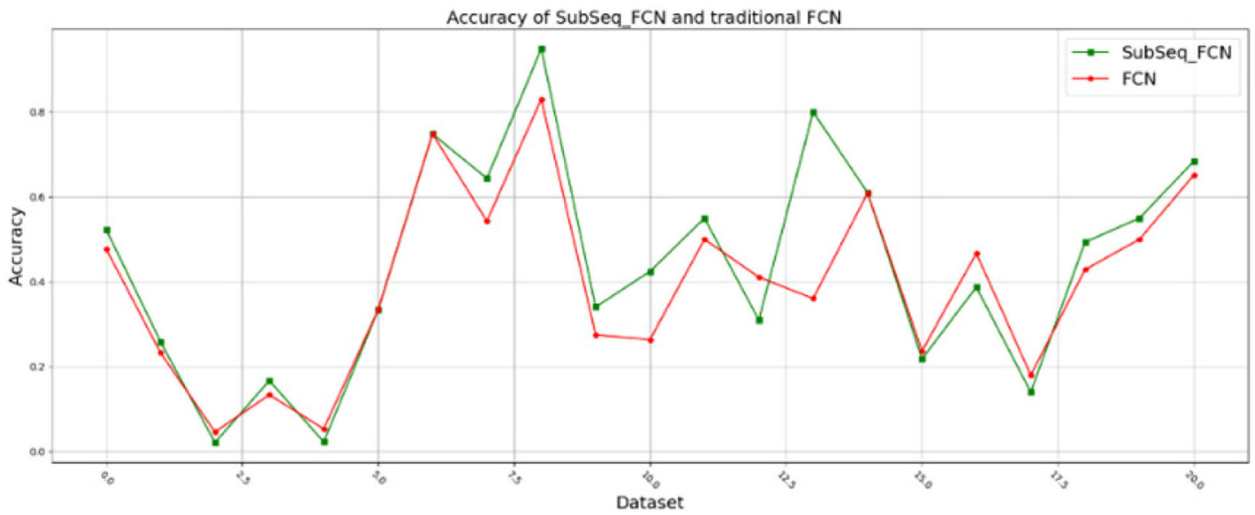

Fig. 13 Accuracy of SubSeq_FCN and traditional FCN

latter several classes are more obvious), and the trend of the two curves in the first few classes is more consistent, and the algorithm runs relatively. In the visual image of the loss of the original sequence and the feature subsequence on the FCN network, from Fig. 10, we can find that the loss of the FCN based on the statistical feature subsequence is generally smaller than that of the original sequence, and the trends of the two curves are relatively consistent.

In order to verify the generalization ability of the method based on statistical features, the same experiments are performed on the residual network and the same analysis. The accuracy, loss, and time of the original sequence and feature subsequence on ResNet are shown in Table 3.

Table 3 shows that the ResNet performs well in the statistical feature subsequence classification. In many time series categories, when the loss function is equivalent, the new sequence based on statistical features is more accurate than the original sequence in ResNet, and the overall performance is better than the original sequence in terms of accuracy or loss. The ResNet network has a short classification time and a small class average error. The loss and accuracy of the original sequence and the new sequence based on statistical features are obtained and visualized. From the visual image of the accuracy on the original sequence and feature subsequence using the ResNet network, Fig. 11 shows that the accuracy of ResNet based on the statistical feature subsequence is still better than that of the original sequence in many classes, but there are some

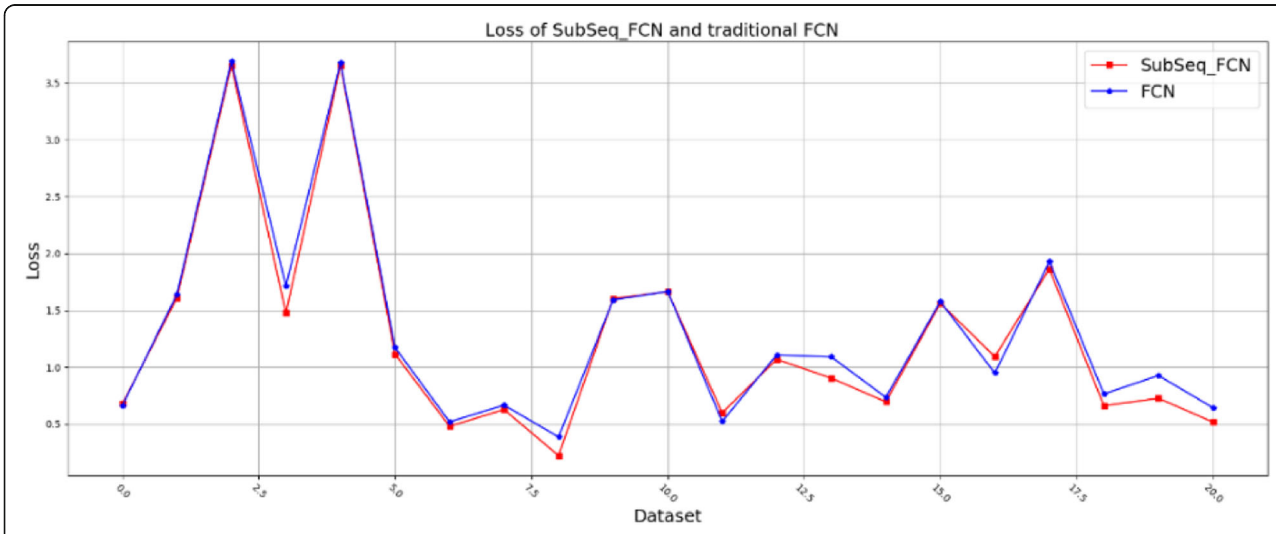

Fig. 14 Loss of SubSeq_FCN and traditional FCN

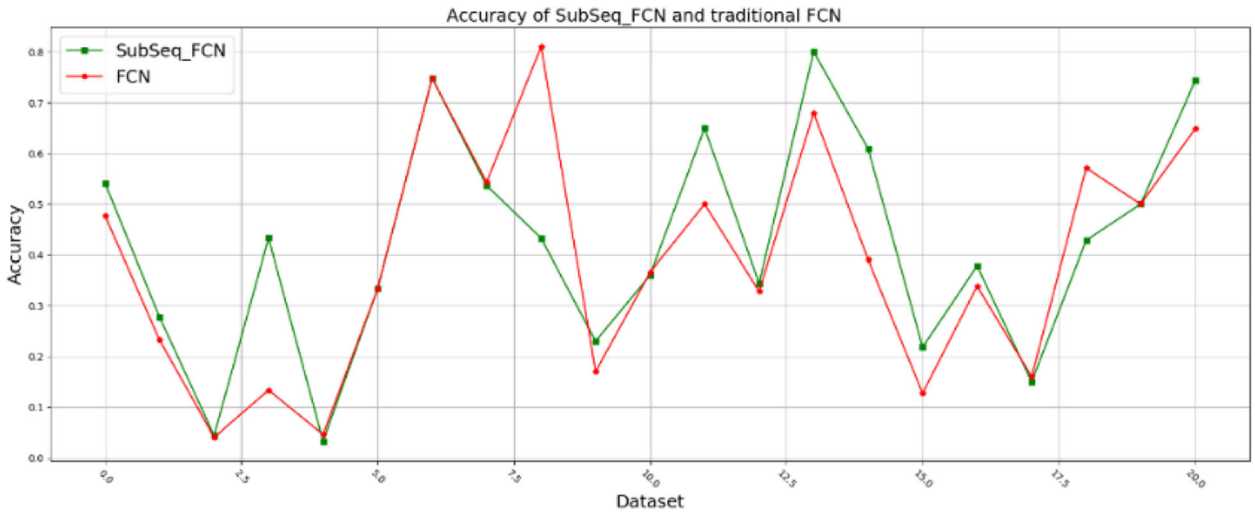

Fig. 15 Accuracy of SubSeq_FCN and traditional FCN

fluctuations in curve changes. In the previous classes, the performance of ResNet based on statistical feature subsequence is improved. After the seventh time series, classes begin to improve. In the visual image of loss degree on the original sequence and feature subsequence, it can be seen that the loss of ResNet network on statistical feature subsequence is generally smaller than that of the original sequence, and the trend of change of the two curves is relatively consistent, as shown in Fig. 12.

5 Improved experiments and result analysis

In this section, we improve the experiments from the following two aspects: increase the number of original sequences and extract more statistical features.

5.1 Increasing the number of original sequence classes

In the previous experiment, we used sequence data from the UCR time series data set, as shown in https://www.cs.ucr.edu/~eamonn/time_series_data_2018/. Fourteen classes were selected from 85 classes with the length of time series exceeding 350 steps. This improvement increased the number of the original time series classes. The number is 22. In this case, the whole convolutional neural network still performs better in the subsequence based on the statistical characteristics. Accuracy and loss are shown in Figs. 13 and 14, respectively.

5.2 Extracting more statistical features

In the data experiment stage, more statistical features can be extracted, such as k-order origin center distance,

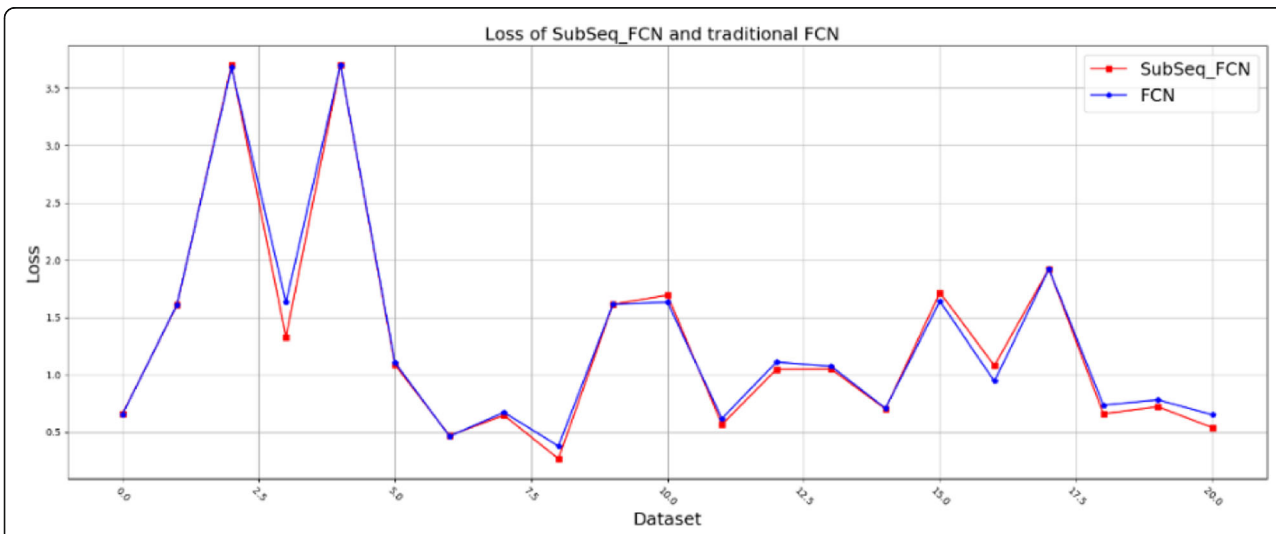

Fig. 16 Loss of SubSeq_FCN and traditional FCN

linear fitting best curve slope, peak value, maximum transform amplitude, and average transform rate. Then we provide on the contrast experiment from the accuracy and the loss aspect, as shown in Figs. 15 and 16.

From Figs. 15 and 16, although the 9 and 19 subsequences are not as accurate as the original sequences, the accuracy of the subsequences on most classes is improved with the addition of some features.

6 Conclusions

FCN has been shown to achieve state-of-the-art performance on the task of TSC. Based on FCN, this paper studies the time series preprocessing and found a way to improve the performance of FCN classification. The method uses the window slicing principle to divide the original time series into multiple equal-length subsequences, then extracts statistical features on each subsequence to form a new sequence as the input of the neural network, adds the fine-tuning idea in the network training process, and then evaluates the classification performance of the network by using test set. Finally, the sample sequence complexity and network classification loss accuracy are compared with the FCN using the original sequence. It is proved that the FCN performs better on the samples of the statistical features of the extracted subsequences. In order to prove that the proposed method has generalization ability to the network structure and the same experiment and analysis on the ResNet, the method based on statistical feature subsequence also improves the classification performance of ResNet.

Several problems remain unsolved. Some of the interesting problems are how to adjust subsequence length/step length and regularize a network and how to use LSTM-FCN, ALSTM-FCN [9], hierarchies of feature [19], or time aware [20, 21] to improve the performance of time series classification. In addition, we will try to apply the method of classification service in distributed cloud/fog environment [22–32] in the future.

Abbreviations

DTW: Dynamic time warping; FCN: Fully convolutional neural networks; KNN: *K*-nearest neighbor; MPCE: Mean per class error; OTS: Original time series; SEF: Sequence of extracted features; SVM: Support vector machines; TSC: Time series classification

Acknowledgements

Zhang Linkun and Wang Zhengyan help us read through the text and polish some sentences. The authors thank the anonymous reviewers for their valuable suggestions and contributions to the quality of the paper.

Authors' contributions

YL and ZW designed the study. ZW performed the simulation experiments and wrote the paper. YL and ZW reviewed and edited the manuscript. Both authors read and approved the final manuscript.

Authors' information

Yuxia Lei obtained his Ph.D. degree in computer software and theory from the Institute of Computing Technology, CAS, in 2010. He is an associate professor. His current research interests include concept lattice, artificial intelligence, and machine learning. Zhongqiang Wu received the B.Sc. degree in computer science from Qufu Normal University, Rizhao, China, in 2019.

Funding

This work is partly supported by the Natural Science Foundation of China (under grant no. 61502272), the Promotive Research Fund for Excellent Young and Middle-Aged Scientists of Shandong Province (under grant no. BS2014DX005), and Undergraduate Education Reform Project in Shandong Province (no. Z2018S022).

Availability of data and materials

The original sequence data set comes from the UCR time series and the data can be found in https://www.cs.ucr.edu/~eamonn/time_series_data_2018/.

Competing interests

The authors declare that they have no competing interests.

Received: 12 September 2019 Accepted: 30 January 2020
Published online: 19 February 2020

References

1. Z. Cui, W. Chen, Y. Chen, Multi-scale convolutional neural networks for time series classification [J]. *CoRR* **1603**(06995) (2016)
2. Z. Wang, W. Yan and T. Oates. Time series classification from scratch with deep neural networks: a strong baseline [J]. *International Joint Conference on Neural Networks* (IJCNN). 2017, 1578-1585.
3. J. Lin, E. Keogh, L. Wei, S. Lonardi, Experiencing SAX: a novel symbolic representation of time series [J]. *Data Mining and knowledge discovery* **15**(2), 107–144 (2007)
4. M.G. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **35**(11), 2796–2802 (2013)
5. P. Schäfer. The boss is concerned with time series classification in the presence of noise [J] *Data Mining and Knowledge Discovery*, 2015, 29(6): 1505-1530.
6. P. Schäfer, Scalable time series classification [J]. *Data Mining and Knowledge Discovery* **30**(5), 1273–1298 (2016)
7. J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures [J]. *Data Mining and Knowledge Discovery* **29**(3), 565–592 (2015)
8. A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with COTE: the collective of transformation-based ensembles [J]. IEEE Transactions on Knowledge and Data Engineering **27**(9), 2522–2535 (2015)
9. F. Karim, S. Majumdar, H. Darabi, S. Chen, LSTM fully convolutional networks for time series classification [J]. *IEEE Access*., 1662–1669 (2018)
10. Y. Chen, E. Keogh, B. Hu, N. Begum, A. Bagnall, A. Mueen, and G. Batista, The UCR time series classification archive [Online], 2015, www.cs.ucr.edu/~eamonn/time_series_data/
11. H I Fawaz, G. Forestier, J. Weber, et al. Deep learning for time series classification: a review [J]. *Data Mining and Knowledge Discovery*, 2019, 33(4): 917-963, 2019.
12. C. Lea, R. Vidal, A. Reiter, and G. D. Hager, Temporal convolutional networks: a unified approach to action segmentation [C]. ECCV 2016: Computer Vision – ECCV 2016 Workshops, 2016: 47-54.
13. J. R. Chang, Y.S. Chen. Batch-normalized maxout network in network [J]. Computer Science, 2015.
14. He K, Zhang X, Ren S, et al. Deep residual learning for image recognition [C]//Proceedings of the IEEE conference on computer vision and pattern recognition. 2016: 770-778.
15. S. Loffe and C. Szegedy. Batch normalization: accelerating deep network training by reducing internal covariate shift [J], *International Conference on Machine Learning*, pp.448-456, 2015.
16. N. Srivastava, G.E. Hinton, A. Krizhevsky, I. Sutskever, R. Salakhutdinov, Dropout: a simple way to prevent neural networks from overfitting [J]. *Journal of Machine Learning Research* **15**(1), 1929–1958 (2014)
17. Y. Zheng, Q. Liu, E. Chen, Y. Ge, J.L. Zhao, Exploiting multi-channels deep convolutional neural networks for multivariate time series classification [J]. *Frontiers of Computer Science* **10**(1), 96–112 (2016)

18. J. Long, E. Shelhamer, and T. Darrell, Fully convolutional networks for semantic segmentation, in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3431-3440.
19. Y. Lei , Y. Yan, Y. Han, and F. Jiang. The hierarchies of multivalued attribute domains and corresponding applications in data mining [J]. *Wireless Communications and Mobile Computing*, Volume 2018, Article ID 1789121, 1-11, 2018.
20. L.Y. Qi, R.L. Wang, S.C. Li, Q. He, X.L. Xu, C.H. Hu, Time-aware distributed service recommendation with privacy-preservation [J]. *Information Sciences* **480**, 354–364 (2019)
21. L. Y. Qi, X.Y. Zhang, W.C Dou, C.H. Hu, C. Yang, J.J. Chen. A two-stage locality-sensitive hashing based approach for privacy-preserving mobile service recommendation in cross-platform edge environment [J]. *Future Generation Computer Systems*, 2018, 88:636-643.
22. H.W. Liu, H.Z. Kou, C. Yan, L.Y. Qi, Link prediction in paper citation network to construct paper correlation graph [J]. *EURASIP Journal on Wireless Communications and Networking* (2019). https://doi.org/10.1186/s13638-019-1561-7
23. W. W. Gong, L.Y. Qi, Y.W. Xu. Privacy-aware multidimensional mobile service quality prediction and recommendation in distributed fog environment [J]. *Wireless Communications and Mobile Computing*, vol. 2018, Article ID 3075849, 8 pages, 2018.
24. Y. W. Xu, L.Y. Qi, W.C. Dou, J.G. Yu. Privacy-preserving and scalable service recommendation based on SimHash in a distributed cloud environment [J]. *Complexity*, Volume 2017, Article ID 3437854, 9 pages, 2017.
25. X. K. Wang, Laurence T. Yang, L.W. Kuang, X.G. Liu, Q.X. Zhang and M. Jamal Deen, A tensor-based big-data-driven routing recommendation approach for heterogeneous networks [J], *IEEE Network Magazine*, 2019, 33(1): 64-69.
26. X.K. Wang, Laurence T. Yang, H.G. Li, M. Lin, J.J. Han, Bernady O. Apduhan, NQA: a nested anti-collision algorithm for RFID systems [J]. *ACM Transactions on Embedded Computing Systems* **18**(4) (2019). https://doi.org/10.1145/3330139
27. X. K. Wang, Laurence T. Yang, Y.H. Wang, X.G. Liu, Q.X. Zhang, and M. Jamal Deen, A distributed tensor-train decomposition method for cyber-physical-social services [J], *ACM Transactions on Cyber-Physical Systems*, 3(4), doi: https://doi.org/10.1145/3323926, 2019.
28. X.K. Wang, L.T. Yang, X. Xie, J.R. Jin, M.J. Deen, A cloud-edge computing framework for cyber-physical-social services [J]. *IEEE Communications Magazine* **55**(11), 80–85 (2017)
29. X.L. Xu, R.C. Mo, F. Dai, W.M. Lin, S.H. Wan, W.C. Dou, Dynamic resource provisioning with fault tolerance for data-intensive meteorological workflows in cloud [J]. *IEEE Transactions on Industrial Informatics* (2019). https://doi.org/10.1109/TII.2019.2959258
30. X.L. Xu, X.H. Liu, Z.Y. Xu, C.J. Wang, S.H. Wan, X.X. Yang, Joint optimization of resource utilization and load balance with privacy preservation for edge services in 5G networks [J]. *Mobile Networks and Applications* (2019). https://doi.org/10.1007/s11036-019-01448-8
31. X.L. Xu, S.C. Fu, W.M. Li, F. Dai, H.H. Gao, V. Chang, Multi-objective data placement for workflow management in cloud infrastructure using NSGA-II [J]. IEEE Transactions on Emerging Topics in Computational Intelligence (2019)
32. X. L. Xu, C. X. He, Z. Y. Xu, L. Y. Qi, S.H. Wan, MZA Bhuiyan. Joint optimization of offloading utility and privacy for edge computing enabled IoT [J]. IEEE Internet of Things Journal, 2019, Doi: 10.1109/JIOT.2019.2944007.

Publisher's Note