The Journal of Supercomputing (2021) 77:7021–7045
https://doi.org/10.1007/s11227-020-03560-z

Bidirectional LSTM-RNN-based hybrid deep learning frameworks for univariate time series classification

Mehak Khan[1] · Hongzhi Wang[1] · Adnan Riaz[2] · Aya Elfatyany[1] · Sajida Karim[1]

Accepted: 3 December 2020 / Published online: 4 January 2021

Abstract

Time series classification (TSC) has been around for recent decades as a significant research problem for industry practitioners as well as academic researchers. Due to the rapid increase in temporal data in a wide range of disciplines, an incredible amount of algorithms have been proposed. This paper proposes robust approaches based on state-of-the-art techniques, bidirectional long short-term memory (BiLSTM), fully convolutional network (FCN), and attention mechanism. A BiLSTM considers both forward and backward dependencies, and FCN is proven to be good at feature extraction as a TSC baseline. Therefore, we augment BiLSTM and FCN in a hybrid deep learning architecture, BiLSTM-FCN. Moreover, we similarly explore the use of the attention mechanism to check its efficiency on BiLSTM-FCN and propose another model ABiLSTM-FCN. We validate the performance on 85 datasets from the University of California Riverside (UCR) univariate time series archive. The proposed models are evaluated in terms of classification testing error and f1-score and also provide performance comparison with various existing state-of-the-art techniques. The experimental results show that our proposed models perform comprehensively better than the existing state-of-the-art methods and baselines.

Keywords Deep learning · Time series classification · Bidirectional long short-term memory recurrent neural network · Convolutional neural network · Attention mechanism

1 Introduction

Time series is a sequence of real-valued data points measured continuously at unvarying time intervals. TSC has been a significant research problem in the past decades. TSC is a task to allocate unlabeled time series data to pre-defined classes. The ubiquitous nature of time series data directs the scope of applications

✉ Mehak Khan
mehakkhan@hit.edu.cn

Extended author information available on the last page of the article concurrently with the development of more matured and practical solutions to deal with problems of improving performance and computational complexity [1, 2]. A successful TSC model can capture and generalize the pattern of time series signals such that it can classify unseen data [3]. Also, the diversity of the dataset's types in the univariate time series UCR archive [4, 5] shows the diverse applications of the TSC problem. Several methods have been applied to TSC, including distance-based, feature-based, ensemble-based, and deep neural networks.

In distance-based methods, the key is to measure the similarity between the given time series. Based on similarity metrics, nearest neighbor (NN) or support vector machine (SVM) with similarity-based kernels can be used for TSC. As the most primitive baseline, distance-based methods work directly on a raw time series with some pre-defined similarity measures such as Euclidean distance (ED) or dynamic time warping (DTW) [6] to perform classification. One of the most common and traditional TSC approaches is the use of the nearest neighbor classifier, coupled with DTW (1NN-DTW) [7], and this approach is also known as a benchmark classifier.

In feature-based methods, each time series is converted to a set of global features, which is used to define the similarity between pairs of time series. Feature-based representations of time series can be used to tackle a wide range of TSC problems in a way that provides interpretability, with the choice of feature-based representation determining the types of insights that can be gained about the problem at hand. Bag of words (BoW) [8], bag of features (TSBF) [9], bag of SFA symbols (BOSS) [10], 1NN bag of SFA symbols (1NN-BOSS) [10], bag of SFA symbols in vector space (BOSS VS) [11], and Word ExtrAction for time SEries cLassification (WEASEL) [12] have achieved outstanding performance.

The ensemble-based method combines different classifiers into a single classifier to achieve higher accuracy and efficiency. The elastic ensemble (PROP) [13] is a combination of 11 classifiers based on elastic distance measures, including DTW CV, DTW, LCSS, and ED with a weighted ensemble scheme. The flat collective of transformation-based ensembles (COTE) [14] is an ensemble method that combines 35 different classifiers based on the features extracted from time and frequency domains and considered to be the most accurate classifier.

All the methods mentioned above need heavy-hand crafting on data pre-processing and feature engineering [15]. To solve these issues, in the last few years, deep neural networks have been utilized for classifying time series, as a deep neural network does not need heavy feature engineering, and data pre-processing. Multi-scale convolutional neural network (MCNN) [16], a convolutional neural network designed explicitly for classifying time series, down sampling, skip sampling, and sliding windows were used for pre-processing the data to prepare for the multi-scale settings manually. Multilayer perceptron (MLP), FCN, and residual network (ResNet) [15] provided a simple and robust baseline for TSC from scratch and out-performed on many UCR archive datasets without any heavy pre-processing and feature engineering. Karim et al. [17] presented two hybrid models, LSTM-FCN and ALSTM-FCN, using well-known deep learning methods, attention mechanism, long short-term memory (LSTM), and FCN. They exploited LSTM to improve the performance of FCN by augmenting the FCN module with either LSTM or attention LSTM. These models showed better results than FCN and several state-of-the-art methods on a majority of the UCR archive datasets. Later, LSTM-FCN architecture is employed to solve various time series-related problems [18–20].

In this paper, our goal is to support efficient analysis for TSC; therefore, we propose two hybrid models for end-to-end TSC based on BiLSTM, FCN, and attention mechanism. The combination of BiLSTM and FCN has been employed previously for different applications, such as breast cancer [21] and invoice recognition & processing [22]. However, to the best of our knowledge, no prior work and experimental studies were found about the efficiency of BiLSTM and FCN for the univariate TSC problem. Besides, we utilize the attention mechanism on BiLSTM. Furthermore, we also studied the dropout [23] technique and investigated proposed models with its absence and presence to provide a better illustration.

Our main contribution in this paper is threefold:

1. We propose two hybrid deep learning models; BiLSTM with FCN (BiLSTM-FCN) and attention-based BiLSTM with FCN (ABiLSTM-FCN).
2. The study shows the augmentation of state-of-the-art techniques, BiLSTM, FCN, and attention mechanism in hybrid models; BiLSTM-FCN and ABiLSTM-FCN univariate TSC problem. We also explore the usage of the dropout technique.
3. The experimental results demonstrate a performance comparison with existing literature and validate that the proposed models are end-to-end and do not require heavy data pre-processing, feature engineering, and refinement.

The rest of the paper is structured as follows: Sect. 2 presents the methodology with problem formulation and background components, Sect. 3 briefly explains experiments, Sect. 4 demonstrates results and discussions, and we conclude our work in Sect. 5.

2 Methodology

In this section, we first formulate the problem and then describe the architecture of the proposed models with their background components.

2.1 Problem formulation

Definition 1 A univariate time series U is an ordered set of these real values: $U = (x_1, x_2, ..., x_N)$. The dimension U is equal to the number of real values N.

Definition 2 A dataset $D = \{(a_1, b_1), (a_2, b_2), ..., (a_N, b_N)\}$ is a group of pairs (A_i, B_i) where A_i is a univariate time series with B_i to its corresponding one hot label vector.

Definition 3 The univariate TSC problem is defined as follows: given a set of classes Y, a training data τ of time series U_i associated with their class labels $y(U_i) \in Y$, i.e.,

$\tau = \{(U_1, y(U_1)), ..., (U_m, y(U_m))\}$, the goal is to find a function f, which is a classifier or model, so that $f(U) = y(U)$ and for time series $U \notin \tau$ [24].

2.2 BiLSTM-FCN and ABiLSTM-FCN

In this study, we propose BiLSTM-FCN & ABiLSTM-FCN by augmenting BiLSTM, FCN, and attention mechanism in two robust hybrid models.

2.2.1 BiLSTM

Using LSTM as the network architecture in a bidirectional recurrent neural network (BRNN) yields BiLSTM. Combining the advantages of BRNN and LSTM, BiLSTM-based recurrent neural networks (RNN) were designed [25].

BRNN was first introduced by [26] to present a structure that unfolds to become a bidirectional neural network. When it is applied to time series data, not solely the information can be passed following the natural temporal sequences, but additional information can also reversely provide information to previous time steps. BRNN comprises of two hidden layers; both hidden layers are connected to input and output. These layers are differentiated in the way that the first has recurrent connections from past time step, while the second is flipped, passing activation backward on the sequence. BRNN can be trained by regular backpropagation after unfolding across time. The subsequent three equations describe a BRNN [27]:

$$h^{(t)} = \sigma(W^{hx} x^{(t)} + W^{hh} h^{(t-1)} + b_h) \tag{1}$$

$$z^{(t)} = \sigma(W^{zx} x^{(t)} + W^{zz} z^{(t+1)} + b_z) \tag{2}$$

$$\hat{y}^{(t)} = \text{softmax}(W^{yx} h^{(t)} + W^{yx} z^{(t)} + b_y) \tag{3}$$

where $h^{(t)}$ and $z^{(t)}$ are the values of the hidden layers in the forward and backward directions, respectively. At the current time step t, recurrent edges receive input from current data point $x^{(t)}$ and the previous state $h^{(t-1)}$, whereas W and b represent weight matrix and bias vectors. The σ is the sigmoid function, and softmax works as an activation function at the output layer $\hat{y}^{(t)}$.

Hochreiter and Schmidhuber introduced the LSTM primarily to overcome the vanishing gradient problem [28]. LSTM is a variant of RNN that has an identical type of input and output, as shown in Fig. 1a. However, in distinction to RNN, LSTM has an input gate, a forget gate, and an output gate. Therefore, it can control what has to be kept and what has to be forgotten. That is why LSTM can hold information from the past, whereas RNN cannot [29].

Put formally, computation within the LSTM model yields in keeping with the subsequent calculations that are performed at each time step. These calculations offer the complete algorithm for a modern LSTM with forget gates [27]:

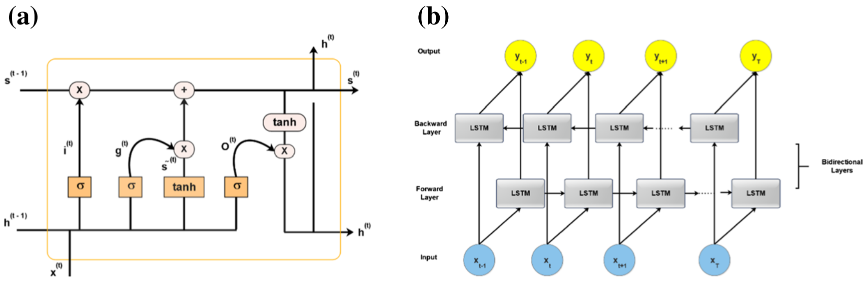


Fig. 1 **a** LSTM architecture, **b** BiLSTM architecture

$$g^{(t)} = \tanh(W^{gx}x^{(t)} + W^{gh}h^{(t-1)} + b_g) \tag{4}$$

$$i^{(t)} = \sigma(W^{ix}x^{(t)} + W^{ih}h^{(t-1)} + b_i) \tag{5}$$

$$f^{(t)} = \sigma(W^{fx}x^{(t)} + W^{fh}h^{(t-1)} + b_f) \tag{6}$$

$$o^{(t)} = \sigma(W^{ox}x^{(t)} + W^{oh}h^{(t-1)} + b_o) \tag{7}$$

$$s^{(t)} = g^{(t)} \odot i^{(i)} + s^{(t-1)} \odot f^{(t)} \tag{8}$$

$$h^{(t)} = \tanh(s^{(t)}) \odot o^{(t)} \tag{9}$$

where $x^{(t)}$ is the input layer at the current time step t, $h^{(t)}$ is the value of the hidden layer of LSTM, while $h^{(t-1)}$ denotes output values by each memory cell in the hidden layer at the previous time. The σ represents a sigmoid function, $\odot$ and tanh is element-wise multiplication and hyperbolic tangent function, respectively.

BiLSTM processes sequence data in each forward and backward direction with two separate hidden layers to capture past and future information, respectively; then, the two hidden states are concatenated to produce the final output like as shown in Fig. 1b. It has been proved that bidirectional networks are considerably better than unidirectional ones in many fields. We use BiLSTM because it provides access to the long-range context in both input directions and outcomes with full learning on the particular problem. Unidirectional LSTM processed data based on the preserved information solely from the past. In issues where all time steps of the input sequences are available, BiLSTM trains two instead of one LSTMs on the input sequence. BiLSTM has achieved state-of-the-art results on phoneme classification [25], handwriting recognition [30], natural language processing [31], speech recognition [32], and many more.

2.2.2 FCN

Convolutional neural networks (CNNs) are influential graphic models that have the ability to yield hierarchies of features. FCN is an extension of classical CNNs that were primarily proposed by Wang et al. [15] for TSC and validated on the UCR archive. FCNs are mostly applied in the temporal domain and have ended up to be useful for dealing with the temporal dimension for TSC without any immense data pre-processing and feature engineering. In the proposed models, we use FCN as a feature extractor in the first branch of both models.

For univariate TSC, FCN is described as follows:

$$t = w \odot x + b \tag{10}$$

$$a = \mathrm{BN}(t) \tag{11}$$

$$y = \mathrm{ReLU}(a) \tag{12}$$

where w, x, and b represent tensor, input vector, and bias vector at time step t, respectively, and $\odot$ is the convolution operator. The FCN architecture consists of three convolutional 1D kernels (8, 5, and 3) with filter sizes of 128, 256, and 128 in each block without striding. Each block is followed by a batch normalization (BN) [33] and a rectified linear unit (ReLU) activation layer [34]. After the convolutional blocks, features are feed into a global average pooling layer [35], and then, the final label y is produced from a SoftMax layer.

2.2.3 Attention mechanism

Attention models are lightly based on a bionic design to simulate the behavior of human vision when humans look at an image; we do not scan it bit by bit or stare at the entire image. However, we focus on some critical parts of it and gradually build the context after catching the idea. The attention models typically refer to the models that were introduced for machine translation [36] and soon applied to many different domains such as for speech recognition [37], image caption generation [38], emotion classification [39], and time series classification [17, 40].

As discussed in [41], it computes a context vector c_t as the linear combination of a sequence of RNNs latent vector h:

$$c_t = \sum_{i=1}^{n} \alpha_{t,i} h_i \tag{13}$$

$$\alpha_{t,i} = \frac{\exp(e_{t,i})}{\sum_{i=1}^{n} \exp(e_{t,i})} \tag{14}$$

where n is an input sequence length, and $\alpha_{t,i}$ is a normalized weight calculated by applying a SoftMax function on an attention weight $e_{t,i}$. An attention weight is a learnable function of input at the current time step h_i; additionally, a previous cell state s_{t-1} of the decoder $e_{t-1} = \alpha(s_{t-1}, h_i)$.

2.2.4 Augmenting BiLSTM and attention-based BiLSTM with FCN

In this paper, we propose two deep learning models; bidirectional long short-term memory with FCN (BiLSTM-FCN) and attention-based bidirectional long short-term memory with FCN (ABiLSTM-FCN). The properties of these models are summarized in Table 1.

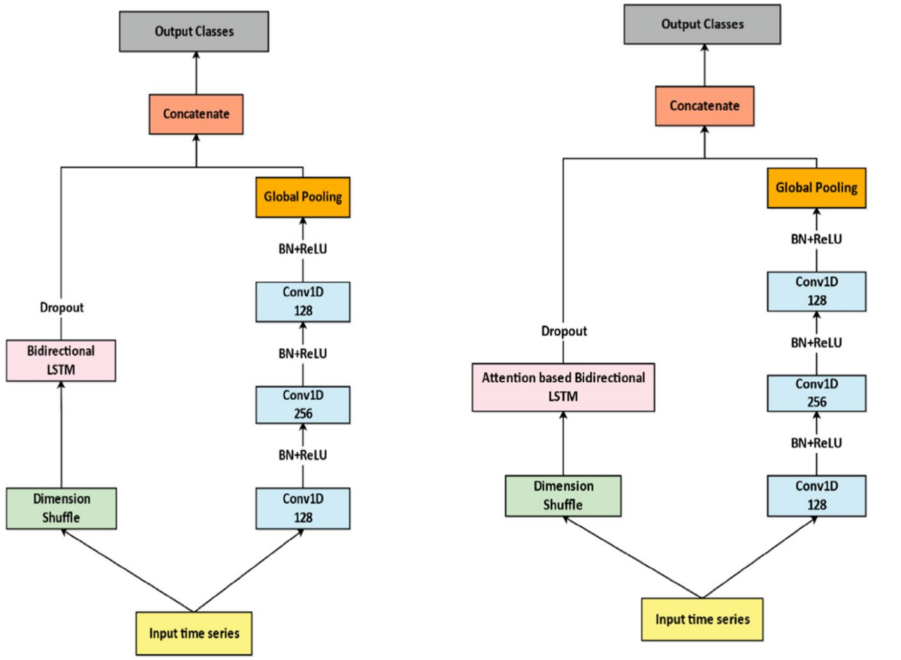
We augment BiLSTM, FCN, and attention mechanism to propose end-to-end hybrid models since a BiLSTM considers both forward and backward dependencies in time series data. Whereas FCN has the ability to yield hierarchies of features, the utilization of the attention mechanism allows the network to focus on the ardently salient parts of a time series data. Therefore, these components can show superior performance for the univariate TSC task.

The proposed model architecture consists of two branches, FCN, and BiLSTM or ABiLSTM, respectively, as depicted in Fig. 2a, b. The first branch is the convolutional part of the model used as a feature extractor; this block has three convolutional 1D kernels with the sizes (8, 5, and 3) without striding. Each layer is followed by batch normalization to improve the speed, performance, and stability of the network and a ReLU activation layer to fix the vanishing gradient problem in a deep neural network, and build the final network by stacking three convolution blocks with the filter sizes 128, 256, and 128 in each block. After the convolution blocks, the features are feed into a global average pooling layer.

The second branch consists of a bidirectional long short-term memory (BiLSTM) block or an attention-based bidirectional long short-term memory (ABiLSTM). BiLSTM trains two LSTMs instead of one, which can access long-range context in both forward and backward directions in time series data. BiLSTM or ABiLSTM block receives a transformed time series after dimension shuffle layer and then transposes the temporal dimension of the time series, followed by dropout, to avoid overfitting, and finally, the output of two branches is concatenated and feed on to a SoftMax classifier.

Table 1 The network properties of proposed models

Network properties	
Model type	Hybrid deep neural network
Components	BiLSTM, FCN, attention mechanism
Input data type	Univariate time series
Activation function used in FCN	ReLU
Activation function at the output layer	SoftMax
Optimizer	Adam with 0.001 learning rate
Evaluation metrics	Classification testing error, f1-score



(a) Bidirectional Long Short-Term Memory and Fully Convolutional Network (BiLSTM-FCN)

(b) Attention-based Bidirectional Long Short-Term Memory and Fully Convolutional Network (ABiLSTM-FCN)

Fig. 2 The network architectures of proposed models

The input to the network is a raw univariate time series dataset from the UCR archive in both branches, consecutively. FCN is responsible for extracting features from the data, while BiLSTM or ABilSTM block is taking input from the dimension shuffle layer, which processes it from both forward and backward direction. The BiSLTM parameters will significantly increase to classify the class labels, and the increase in a number of parameters can also raise the exertion of network optimization, which can affect the performance of models. Therefore, we use the dimension shuffle [17] layer to provide input after transforming the univariate time series dataset into multivariate time series with a single time step. It improves the efficiency of the network to train a model in less time and augments the performance of FCN. The FCN is good at extracting useful features from the input time series, and the BiLSTM is good at learning with the temporal structure of the input sequence since it uses contextual information from both past and future of its input sequences. Attention-based BiLSTM summarizes the hidden states for the output and also strengthens the results by aggregating the hidden states and weighing their relative importance.

Models were trained by using the Adam optimizer [42] with an initial learning rate of 0.001 and reduced to the minimum learning rate 0.0001. He uniform initializer was used to initialize weights and ReLU activation function in all the three convolutional kernels.

3 Experiments and results

3.1 Datasets

For the experiments, we evaluated proposed models on the 2015 UCR time series classification archive [4]. It is a diverse collection of 85 univariate time series datasets, and all the datasets are already z-normalized to remove offset and scaling. Transformed data have zero mean and a unit of standard deviation [5]. They do not need further data pre-processing. Each dataset is split into train and test sets by default [4]; we remain dataset unchanged to make our results comparable with the prior methods. The number of classes differs from 2 to 60; sequence length is between 24 and 2,709 observations; and the test set is mostly larger than the training set. It also contains different types of collected sources of datasets such as Image, Sensor, Motion, Spectro, Device, ElectroCardioGram (ECG), and Simulated.

3.2 Experimental settings

We demonstrate the performance of our proposed models on 85 datasets from the UCR archive. Throughout the experiments, the settings are kept constant for BiLSTM-FCN and ABiLSTM-FCN models to have a fair comparison with present state-of-the-art methods. Since datasets are already z-normalized, the proposed models do not need any further data pre-processing. The models were trained with the batch size range of 8–128 cells. The training epochs were set as 2000 for most of the datasets but increased according to the sequence length of the dataset to prevent overfitting. The proposed models were experimented multiple times to acquire the best performance using a single GPU GTX 1060, Keras library [43] with the TensorFlow [44] in the backend.

3.3 Dropout training

The deep neural networks are difficult to train, and overfitting is one of the major challenges. We applied dropout rates 0.8 to the input layer of BiLSTM and ABiLSTM block to prevent overfitting and improve regularization. Table 2 shows the performance of two datasets with the absence and presence of dropout. It is evident that the dropout generally improves regularization compared to the networks which do not use dropout and also gives significant improvements in performance.

Table 2 Performance with the presence and absence of dropout in proposed models

Dataset	Type	Model	With dropout (0.8)	Without dropout
Worms Two Class	Motion	**BiLSTM-FCN**	0.204	0.270
		ABiLSTM-FCN	0.198	0.248
Swedish Leaf	Image	**BiLSTM-FCN**	0.019	0.027
		ABiLSTM-FCN	0.017	0.023

3.4 Evaluation metrics

In this study, we evaluated BiLSTM-FCN and ABiLSTM-FCN and other comparative approaches using classification testing error rate, f1-score, arithmetic mean rank, and mean per class error (MPCE) (lower is better) as shown in Table 3.

The classification testing error loss is the average error measured by calculating the error on data that was unknown during the training phase. The testing error is also known as a generalization error. The testing error loss is used as evaluation metrics for various applications, including natural language processing and time series classification [15, 45].

The f1-score conveys an inclusive measure of accuracy by combining precision and recall. There are numerous applications where f1-score is used as evaluation metrics to measure a model's performance, i.e., information retrieval and natural language processing [46].

The f1-score can be calculated as follows:

$$\text{precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{15}$$

$$\text{recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{16}$$

$$f1 - \text{score} = 2 \times \frac{\text{precision} \times \text{recall}}{\text{precision} + \text{recall}} \tag{17}$$

where TP*,* FP*, and* FN are defined as true-positive, false-positive, and false-negative, respectively.

MPCE is an evaluation metric introduced by [15] to evaluate the performance of the specific classifier on multiple datasets. MPCE is the arithmetic mean of PCE, which is calculated as follows:

$$\text{PCE}_n = \frac{e_n}{c_n} \tag{18}$$

$$\text{MPCE} = \frac{1}{N} \sum \text{PCE}_n \tag{19}$$

Table 3 Performance comparison (classification testing error rate) of proposed models with state-of-the-art TSC algorithms on 85 UCR archive time series datasets

Dataset	BiLSTM-FCN	ABiLSTM-FCN	LSTM-FCN	ALSTM-FCN	FCN	ResNet	MLP	MCNN	BOSS-VS	COTE	PROP	BOSS	1NN-BOSS	TSBF	1NN-DTW	DTW	ED
Adiac	**0.122**	0.125	0.140	0.139	0.143	0.174	0.248	0.231	0.302	0.233	0.353	0.235	0.220	0.231	0.396	0.396	0.389
ArrowHead	**0.091**	0.102	0.171	0.171	0.120	0.183	0.177	/	0.171	0.138	0.103	1.660	0.143	0.246	0.337	0.297	0.200
Beef	**0.066**	0.1	0.199	0.166	0.250	0.233	0.167	0.367	0.267	0.133	0.367	0.200	0.200	0.434	0.367	0.367	0.333
BeetleFly	**0**	**0**	**0**	**0**	0.050	0.200	0.150	/	**0**	0.050	0.400	0.100	0.100	0.200	0.300	0.300	0.250
BirdChicken	**0**	**0**	0.090	**0**	0.050	0.100	0.200	/	0.100	0.150	0.350	0.050	**0**	0.100	0.250	0.250	0.450
Car	**0.033**	0.05	0.050	0.050	0.083	0.067	0.167	/	/	/	/	0.167	/	0.217	/	0.267	0.267
CBF	0.001	0.001	0.002	0.004	**0**	0.006	0.140	0.002	0.001	0.001	0.002	0.002	**0**	0.013	0.003	0.003	0.148
ChlorineConC	0.167	0.162	0.197	0.176	0.157	0.172	**0.128**	0.203	0.345	0.314	0.360	0.339	0.34	0.308	0.352	0.352	0.350
CinCECGTorso	0.089	0.097	0.155	0.115	0.187	0.229	0.158	**0.058**	0.130	0.064	0.062	0.125	0.125	0.288	0.349	0.349	0.103
Coffee	**0**	**0**	**0**	**0**	**0**	**0**	**0**	0.036	**0.036**	**0**	**0**	**0**	**0**	**0**	**0**	**0**	**0**
Computers	0.155	0.163	0.191	0.147	0.152	0.176	0.460	/	0.324	0.240	**0.116**	0.244	0.296	0.244	0.300	0.300	0.424
CricketX	0.189	0.187	0.197	0.202	0.185	0.179	0.431	0.182	0.346	**0.154**	0.203	0.259	0.259	0.295	0.246	0.246	0.423
CricketY	0.176	0.171	0.187	0.182	0.208	0.195	0.405	**0.154**	0.328	0.167	0.156	0.208	0.208	0.265	0.256	0.256	0.433
CricketZ	0.171	0.156	0.184	0.184	0.187	0.187	0.408	0.142	0.313	**0.128**	0.156	0.246	0.246	0.285	0.246	0.246	0.413
DiatomSize Reduction	0.042	0.049	0.052	0.063	0.070	0.069	0.036	**0.023**	0.036	0.082	0.059	0.046	0.046	0.102	0.033	0.033	0.065
DistPhalOutAgeGrp	0.135	0.14	**0.132**	0.145	0.165	0.202	0.173	/	0.155	0.229	0.223	0.272	0.180	0.218	0.208	0.230	0.374
DistPhalOutCorr	**0.153**	0.161	0.166	0.176	0.188	0.18	0.190	/	0.282	0.238	0.232	0.252	0.208	0.288	0.232	0.283	0.283
DistalPhalanxTW	**0.18**	0.185	0.185	0.192	0.210	0.260	0.253	/	0.253	0.317	0.317	0.324	0.223	0.324	0.290	0.410	0.367
Earthquakes	**0.161**	0.17	0.177	0.170	0.199	0.214	0.208	/	0.193	/	0.281	0.186	0.186	0.252	0.258	0.281	0.288
ECG200	**0.079**	0.08	0.080	0.090	0.100	0.130	0.080	/	0.180	0.150	/	0.130	0.130	0.16	0.230	0.230	0.120
ECG5000	**0.051**	0.052	0.055	0.054	0.059	0.069	0.065	/	0.110	0.054	0.350	0.059	0.056	0.061	0.250	0.076	0.075
ECGFiveDays	0.011	0.01	0.011	0.009	0.015	0.045	0.030	**0**	**0**	**0**	0.178	**0**	**0**	0.124	0.232	0.232	0.203
ElectricDevices	0.218	0.228	**0.037**	**0.037**	0.277	0.272	0.420	/	0.351	0.230	0.277	0.201	0.282	0.298	0.399	0.399	0.449

Table 3 (continued)

Dataset	BiL-STM-FCN	ABiL-STM-FCN	LSTM-FCN	ALSTM-FCN	FCN	ResNet	MLP	MCNN	BOSS-VS	COTE	PROP	BOSS	1NN-BOSS	TSBF	1NN-DTW	DTW	ED
FaceAll	**0.028**	0.036	0.060	0.045	0.071	0.166	0.115	0.235	0.241	0.105	0.115	0.210	0.210	0.256	0.192	0.192	0.286
FaceFour	0.045	0.045	0.057	0.057	0.068	0.068	0.17	**0**	0.034	0.091	0.091	**0**	**0**	**0**	0.170	0.171	0.216
FacesUCR	0.05	0.046	0.071	0.057	0.052	**0.042**	0.185	0.063	0.103	0.057	0.063	**0.042**	**0.042**	0.134	0.095	0.095	0.231
FiftyWords	0.224	0.243	0.196	**0.176**	0.321	0.273	0.288	0.190	0.367	0.191	0.180	0.301	0.301	0.242	0.310	0.301	0.369
Fish	0.022	0.017	0.017	0.023	0.029	**0.011**	0.126	0.051	0.017	0.029	0.034	**0.011**	**0.011**	0.166	0.177	0.177	0.217
FordA	0.070	**0.069**	0.072	0.073	0.094	0.072	0.231	/	0.096	/	0.182	0.083	0.083	0.15	0.438	0.444	0.335
FordB	**0.081**	0.083	0.088	**0.081**	0.117	0.100	0.371	/	0.111	/	0.265	0.109	0.109	0.402	0.406	0.38	0.394
GunPoint	**0**	**0**	**0**	**0**	**0**	0.007	0.067	**0**	**0**	0.007	0.007	**0**	**0**	0.014	0.093	0.093	0.087
Ham	**0.19**	0.199	0.209	0.228	0.238	0.219	0.286	/	0.286	0.334	/	0.334	0.343	0.239	0.533	0.533	0.400
HandOutlines	0.111	0.125	0.113	0.358	0.224	0.139	0.193	/	0.152	**0.068**	/	0.098	0.13	0.146	0.202	0.119	0.138
Haptics	0.444	0.438	**0.425**	0.435	0.449	0.494	0.539	0.530	0.584	0.481	0.584	0.536	0.536	0.510	0.623	0.623	0.63
Herring	0.234	0.234	0.250	0.265	0.297	0.406	0.313	/	0.406	0.313	**0.079**	0.454	0.375	0.36	0.469	0.469	0.484
InlineSkate	**0.489**	0.492	0.518	0.507	0.589	0.635	0.649	0.618	0.573	0.551	0.567	0.511	0.511	0.615	0.616	0.616	0.658
InsWingbeat-Sound	0.366	0.368	0.342	**0.329**	0.598	0.469	0.369	/	0.43	/	/	0.479	0.479	0.376	0.645	0.643	0.438
ItalyPower Demand	0.03	**0.026**	0.038	0.032	0.030	0.040	0.034	0.030	0.086	0.036	0.039	0.053	0.053	0.117	0.050	0.050	0.045
LargeKitchen App	0.096	0.087	0.090	**0.083**	0.104	0.107	0.520	/	0.304	0.136	0.232	0.235	0.280	0.472	0.205	0.205	0.507
Lightning2	0.163	0.18	0.197	0.213	0.197	0.246	0.279	0.164	0.262	0.164	**0.115**	0.148	0.148	0.263	0.131	0.131	0.246
Lightning7	0.095	**0.068**	0.082	0.178	0.137	0.164	0.356	0.219	0.288	0.247	0.233	0.342	0.342	0.274	0.274	0.274	0.427
Mallat	0.023	0.191	0.019	**0.016**	0.020	0.021	0.064	0.057	0.064	0.036	0.05	0.058	0.058	0.04	0.066	0.066	0.086
Meat	0.100	0.066	0.116	0.033	0.033	0	0.067	/	0.167	0.067	/	0.100	0.100	0.067	0.067	0.067	0.067
MedicalImages	0.193	**0.190**	0.199	0.204	0.208	0.228	0.271	0.260	0.474	0.258	0.245	0.288	0.288	0.295	0.263	0.263	0.316

Table 3 (continued)

Dataset	BiL-STM-FCN	ABiL-STM-FCN	LSTM-FCN	ALSTM-FCN	FCN	ResNet	MLP	MCNN	BOSS-VS	COTE	PROP	BOSS	1NN-BOSS	TSBF	1NN-DTW	DTW	ED
MidPhaOutAgeGrp	0.185	0.194	0.188	0.189	0.232	0.24	0.265	/	0.253	**0.169**	0.474	0.220	0.218	0.186	0.250	0.500	0.481
MidPhalOutlineCor	0.183	**0.143**	0.160	0.163	0.205	0.207	0.240	/	0.350	0.403	0.210	0.455	0.255	0.423	0.352	0.302	0.234
MiddlePhalanxTW	0.360	0.189	0.383	0.373	0.388	0.393	0.391	/	0.414	0.429	0.63	0.455	0.373	0.403	0.416	0.494	0.487
MoteStrain	0.072	0.070	0.078	0.073	**0.050**	0.105	0.131	0.079	0.115	0.085	0.114	0.073	0.073	0.097	0.165	0.165	0.121
NonInvFetECGTho1	0.033	0.029	0.035	0.025	0.039	0.052	0.058	0.064	0.169	0.093	0.178	0.161	0.161	0.158	0.210	0.210	0.171
NonInvFetECGTho2	0.04	0.042	0.038	0.034	0.045	0.049	0.057	0.060	0.118	0.073	0.112	0.101	0.101	0.139	0.135	0.135	0.120
OliveOil	**0.066**	0.066	0.133	0.067	0.167	0.133	0.600	0.133	0.133	0.100	0.133	0.100	0.100	0.167	0.167	0.167	0.133
OSULeaf	0.012	0.012	**0.004**	0.004	0.012	0.021	0.430	0.271	0.074	0.145	0.194	0.012	0.012	0.240	0.409	0.409	0.479
PhalangesOutCorrect	**0.163**	0.167	0.177	0.170	0.174	0.175	0.170	/	0.317	0.194	/	0.229	0.217	0.171	0.272	0.272	0.239
Phoneme	0.652	0.651	0.650	**0.640**	0.655	0.676	0.902	/	0.825	/	/	0.733	0.733	0.724	0.772	0.772	0.891
Plane	**0**	**0**	**0**	**0**	**0**	**0**	0.019	/	/	/	/	/	/	**0**	/	**0**	0.038
ProxPhalOutAgeGrp	0.112	0.112	0.117	**0.107**	0.151	0.151	0.176	/	0.244	0.121	0.117	0.152	0.137	0.128	0.195	0.195	0.215
ProxPhalOutCorrect	**0.044**	0.065	0.065	0.075	0.100	0.082	0.113	/	0.134	0.142	0.172	0.166	0.131	0.152	0.216	0.217	0.192
ProximalPhalanxTW	**0.162**	0.172	0.167	0.173	0.190	0.193	0.203	/	0.248	0.186	0.244	0.200	0.203	0.191	0.263	0.244	0.293
RefrigerationDevices	**0.410**	0.426	0.421	0.429	0.467	0.472	0.629	/	0.488	0.443	0.424	0.498	0.512	0.528	0.536	0.536	0.605
ScreenType	0.351	0.333	0.341	0.327	0.333	**0.293**	0.592	/	0.464	0.411	0.440	0.536	0.544	0.491	0.603	0.603	0.640
ShapeletSim	**0**	0.005	0.011	0.011	0.133	**0**	0.517	/	0.022	**0**	/	**0**	0.044	0.039	0.350	0.350	0.461
ShapesAll	0.079	**0.076**	0.098	0.100	0.102	0.088	0.225	/	0.188	0.095	0.187	0.092	0.082	0.815	0.232	0.232	0.248

Table 3 (continued)

Dataset	BiL-STM-FCN	ABiL-STM-FCN	LSTM-FCN	ALSTM-FCN	FCN	ResNet	MLP	MCNN	BOSS-VS	COTE	PROP	BOSS	1NN-BOSS	TSBF	1NN-DTW	DTW	ED
SmallKitchenApp	0.197	0.194	0.184	0.203	0.197	0.203	0.611	/	0.221	**0.147**	0.187	0.275	0.200	0.328	0.357	0.357	0.659
SonyAIBORobSurf1	**0.014**	0.016	0.018	0.030	0.032	0.015	0.273	0.230	0.265	0.146	0.293	0.321	0.321	0.205	0.275	0.275	0.305
SonyAIBORobSurf2	0.023	0.025	**0.022**	0.025	0.038	0.038	0.161	0.070	0.188	0.076	0.124	0.098	0.098	0.223	0.169	0.169	0.141
StarLightCurves	0.023	0.023	0.024	0.023	0.033	0.025	0.043	0.023	0.096	0.031	0.079	**0.021**	0.021	0.023	0.093	0.093	0.151
Strawberry	**0**	**0**	0.013	0.013	0.031	0.042	0.033	/	0.024	0.030	/	0.025	0.042	0.046	0.060	0.059	0.054
SwedishLeaf	0.019	0.017	0.021	**0.014**	0.034	0.042	0.107	0.066	0.141	0.046	0.085	0.272	0.072	0.085	0.208	0.208	0.211
Symbols	**0.013**	0.014	0.016	**0.013**	0.038	0.128	0.147	0.049	0.029	0.046	0.049	0.032	0.032	0.055	0.05	0.050	0.101
SyntheticControl	0.003	0.003	0.003	0.006	0.010	**0**	0.05	0.003	0.040	**0**	0.010	0.030	0.030	0.007	0.007	0.007	0.120
ToeSegmentation1	0.021	0.017	**0.013**	**0.013**	0.031	0.035	0.399	/	0.031	0.018	0.079	0.062	0.048	0.220	0.228	0.228	0.320
ToeSegmentation2	0.069	0.061	0.084	0.077	0.085	0.138	0.254	/	0.069	0.047	0.085	0.039	**0.038**	0.200	0.162	0.162	0.192
Trace	**0**	**0**	**0**	**0**	**0**	**0**	0.180	**0**	**0**	0.010	0.010	**0**	**0**	0.020	**0**	**0**	0.240
TwoLeadECG	**0**	**0**	0.001	0.001	**0**	**0**	0.147	0.001	0.001	0.015	**0**	0.004	0.016	0.135	0.096	0.096	0.253
TwoPatterns	0.004	0.004	0.003	0.003	0.103	**0**	0.114	0.002	0.015	**0**	0.067	0.016	0.004	0.024	**0**	**0**	0.093
UWaveGestLibAll	0.065	0.076	0.096	0.107	0.174	0.132	**0.046**	/	0.270	0.196	0.199	0.238	0.241	0.170	0.272	0.108	0.052
UwaveGestLibX	0.170	0.171	**0.151**	0.152	0.246	0.213	0.232	0.180	0.364	0.267	0.199	0.241	0.313	0.264	0.366	0.273	0.261
UwaveGestLibY	0.258	0.256	0.233	0.234	0.275	0.332	0.297	0.268	0.336	0.265	0.283	0.313	0.312	**0.228**	0.342	0.366	0.338
UwaveGestLibZ	0.226	0.224	0.203	0.202	0.271	0.245	0.295	0.232	0.098	0.265	0.290	0.312	**0.059**	0.074	0.108	0.342	0.350
Wafer	**0.001**	**0.001**	**0.001**	0.002	0.003	0.003	0.004	0.002	**0.001**	**0.001**	0.003	**0.001**	**0.001**	0.005	0.020	0.020	0.005
Wine	0.129	**0.111**	0.166	**0.111**	**0.111**	0.204	0.204	/	0.296	0.223	/	0.260	0.167	0.389	0.426	0.426	0.389
WordSynonyms	0.34	0.347	0.329	0.332	0.420	0.368	0.406	0.276	0.491	0.266	**0.226**	0.345	0.345	0.312	0.252	0.351	0.382
Worms	0.309	**0.293**	0.298	0.320	0.331	0.381	0.657	/	0.398	0.442	/	0.442	0.392	0.312	0.536	0.416	0.545

Table 3 (continued)

Dataset	BiL-STM-FCN	ABiL-STM-FCN	LSTM-FCN	ALSTM-FCN	FCN	ResNet	MLP	MCNN	BOSS-VS	COTE	PROP	BOSS	1NN-BOSS	TSBF	1NN-DTW	DTW	ED
WormsTwoClass	0.204	0.198	0.215	0.204	0.271	0.265	0.403	/	0.315	0.221	/	**0.169**	0.243	0.247	0.337	0.377	0.390
Yoga	0.087	0.095	0.089	0.095	0.155	0.142	0.145	0.112	0.169	0.113	0.121	**0.081**	**0.081**	0.181	0.164	0.164	0.170
Wins/Ties	**30**	18	13	21	8	11	3	7	6	11	5	12	13	4	3	4	1
MPCE	**0.03020**	0.0306	0.0332	0.0334	0.0391	0.0416	0.0681	0.0124	0.0528	0.0393	0.0425	0.0541	0.0447	0.0598	0.0722	0.0734	0.080
Arith mean rank	**3.0823**	3.3529	4.4588	4.2705	6.9647	7.3764	11.1647	6.7045	10.5662	7.0256	9.2816	8.3690	7.6144	10.6705	12.0963	12.1058	13.8

Table 4 Performance comparison (f1-score) of proposed models with state-of-the-art TSC algorithms on 85 UCR archive time series datasets

Datasets	BiLSTM-FCN	ABiLSTM-FCN	LSTM-FCN	ALSTM-FCN
Adiac	**0.790**	0.766	0.770	0.780
ArrowHead	**0.721**	0.692	0.694	0.695
Beef	0.819	0.847	**0.873**	0.765
BeetleFly	**1.000**	**1.000**	**1.000**	0.949
BirdChicken	**1.000**	**1.000**	**1.000**	**1.000**
Car	**0.960**	0.932	0.952	0.947
CBF	**0.998**	0.996	0.994	0.989
ChlorineConcentration	0.783	0.773	**0.791**	0.767
CinCECGTorso	**0.421**	0.415	0.321	0.375
Coffee	**1.000**	**1.000**	**1.000**	**1.000**
Computers	0.458	0.483	0.914	0.913
CricketX	0.773	0.782	0.782	**0.784**
CricketY	0.785	0.782	**0.786**	0.776
CricketZ	0.791	**0.799**	0.778	0.761
DiatomSizeReduction	**0.943**	0.933	0.926	0.935
DistalPhalanxOutlineAgeGroup	**0.638**	0.624	0.614	0.636
DistalPhalanxOutlineCorrect	0.799	0.806	0.804	**0.813**
DistalPhalanxTW	0.487	**0.489**	0.469	0.479
Earthquakes	**0.614**	0.530	0.466	0.466
ECG200	0.910	**0.916**	0.900	0.909
ECG5000	0.256	0.260	0.251	**0.263**
ECGFiveDays	0.988	0.989	**0.991**	**0.991**
ElectricDevices	**0.198**	0.195	0. 196	0.197
FaceAll	**0.138**	**0.138**	0. 134	0.136
FaceFour	0.915	0.947	**0.949**	**0.949**
FacesUCR	0.899	**0.900**	0.898	0.896
FiftyWords	**0.365**	0.349	0.330	0.353
Fish	0.970	**0.972**	0.964	0.957
FordA	0.929	**0.930**	0.928	0.928
FordB	0.918	0.918	**0.930**	0.929
GunPoint	**1.000**	**1.000**	**1.000**	**1.000**
Ham	**0.808**	0.800	0.788	0.770
HandOutlines	**0.875**	0.857	0.873	0.866
Haptics	0.497	0.498	**0.523**	0.515
Herring	0.705	0.702	**0.722**	0.694
InlineSkate	0.458	0.456	**0.474**	0.446
InsectWingbeatSound	0.407	0.297	**0.432**	0.410
ItalyPowerDemand	0.970	**0.974**	0.970	0.972
LargeKitchenAppliances	**0.414**	0.404	0.407	0.410
Lightning2	**0.836**	0.819	0.767	0.767
Lightning7	0.798	**0.875**	0.833	0.858

Table 4 (continued)

Datasets	BiLSTM-FCN	ABiLSTM-FCN	LSTM-FCN	ALSTM-FCN
Mallat	0.973	**0.975**	0.970	0.971
Meat	0.905	0.933	0.870	**0.973**
MedicalImages	0.718	**0.721**	0.686	0.701
MiddlePhalanxOutlineAgeGroup	0.455	**0.467**	0.347	0.445
MiddlePhalanxOutlineCorrect	0.788	**0.840**	0.821	0.819
MiddlePhalanxTW	**0.353**	0.254	0.314	0.320
MoteStrain	0.922	**0.929**	0.920	0.915
NonInvasiveFetalECGThorax1	0.906	**0.912**	0.908	0.905
NonInvasiveFetalECGThorax2	**0.904**	0.894	0.896	0.894
OliveOil	0.693	0.853	0.611	**0.885**
OSULeaf	0.976	0.985	0.979	**0.988**
PhalangesOutlinesCorrect	**0.816**	0.809	0.803	0.809
Phoneme	0.025	**0.026**	**0.026**	**0.026**
Plane	**0.888**	0.885	**0.888**	0.882
ProximalPhalanxOutlineAgeGroup	0.601	**0.612**	0.594	0.436
ProximalPhalanxOutlineCorrect	**0.941**	0.912	0.904	0.896
ProximalPhalanxTW	**0.535**	0.500	0.504	0.469
RefrigerationDevices	0.217	**0.257**	0.241	0.241
ScreenType	0.306	0.304	0.302	**0.308**
ShapeletSim	**0.855**	0.849	0.842	0.842
ShapesAll	**0.111**	0.108	0.108	0.107
SmallKitchenAppliances	0.343	0.339	0.361	**0.370**
SonyAIBORobotSurface1	**0.985**	0.983	0.974	0.983
SonyAIBORobotSurface2	0.976	0.974	**0.978**	0.977
StarLightCurves	**0.964**	**0.964**	0.961	0.962
Strawberry	**0.865**	**0.865**	0.818	0.818
SwedishLeaf	0.812	**0.815**	0.801	0.811
Symbols	**0.984**	0.983	0.982	0.974
SyntheticControl	0.514	**0.522**	0.516	0.511
ToeSegmentation1	0.719	0.732	0.746	**0.746**
ToeSegmentation2	**0.585**	0.581	0.563	0.577
Trace	0.974	0.977	**0.986**	0.983
TwoLeadECG	**1.000**	**1.000**	0.999	0.999
TwoPatterns	**0.996**	**0.996**	0.989	0.971
UWaveGestureLibraryAll	0.765	**0.792**	0.766	0.754
UWaveGestureLibraryX	**0.695**	0.683	0.654	0.659
UWaveGestureLibraryY	**0.721**	0.601	0.695	0.686
UWaveGestureLibraryZ	**0.768**	0.766	0.739	0.743
Wafer	0.996	**0.997**	0.996	0.996
Wine	0.868	**0.887**	0.887	0.887
WordSynonyms	**0.361**	0.345	0.327	0.345

Table 4 (continued)

Datasets	BiLSTM-FCN	ABiLSTM-FCN	LSTM-FCN	ALSTM-FCN
Worms	0.439	**0.447**	0.423	0.425
WormsTwoClass	0.530	0.520	0.525	**0.542**
Yoga	0.911	0.903	0.906	**0.914**
Wins/Ties	**38**	32	19	17

where e_n refers to error rate, c_n is the number of class labels in a dataset, n refers to each dataset, and N is the total number of datasets tested on a specific model.

We further investigated the performance by comparing the existing state-of-the-art methods with proposed models using a nonparametric statistical hypothesis test, Wilcoxon signed-rank test, Table 4.

4 Results and discussions

In order to evaluate the performance of our proposed models, we show a comparison with the other existing best methods that claim the state-of-the-art results; long short-term memory fully convolutional network (LSTM-FCN) [17], attention long short-term memory fully convolutional network (ALSTM-FCN) [17], fully convolutional network (FCN) [15], residual network (ResNet) [15], multilayer perceptron (MLP) [15], multiscale CNN (MCNN) [16], flat collective of transformation-based ensembles (COTE) [14], elastic ensemble (PROP) [13], bag of SFA symbols (BOSS) [10], BOSS in vector space (BOSSVS) [11], bag of features (TSBF) [9], 1NN-DTW [7], 1NN-BOSS [10], DTW [6], and ED [6]. We trained LSTM-FCN and ALSTM-FCN from scratch to obtain their performance based on the classification testing error rate and f1-score on each dataset.

Table 3 demonstrates the performance comparison in terms of classification testing error rate, arithmetic mean rank, and MPCE. The proposed models BiLSTM-FCN and ABiLSTM-FCN show better performance over 15 existing TSC

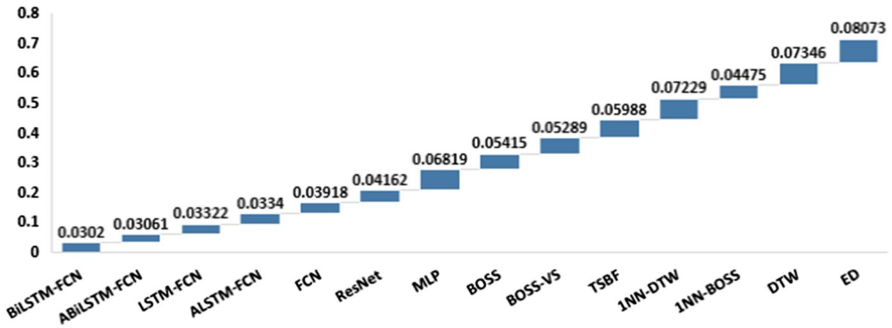


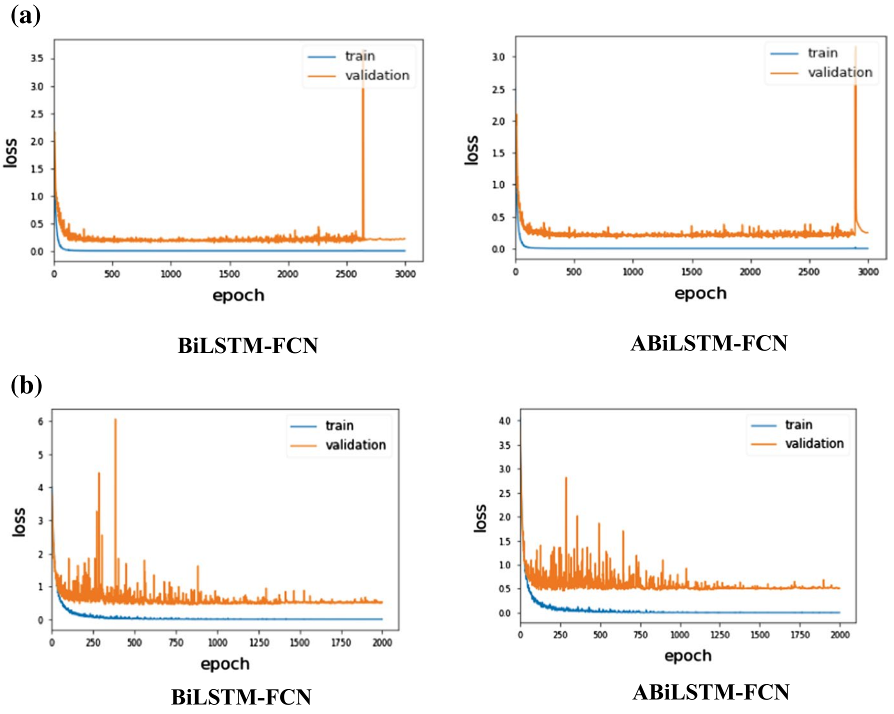
Fig. 3 MPCE score of proposed models with other state-of-the-art methods

Table 5 Wilcoxon signed-rank test comparison

	BiLSTM-FCN	ABiLSTM-FCN	LSTM-FCN	ALSTM-FCN	FCN	ResNet	MLP	MCNN	COTE	PROP	BOSS	BOSS-VS	TSBF	1NN-DTW	1NN-BOSS	DTW
ABiLSTM-FCN	**2.14E-01**															
LSTM-FCN	9.82E-04	**1.47E-03**														
ALSTM-FCN	**9.93E-03**	**8.32E-03**	**3.78E-01**													
FCN	1.65E-11	1.18E-11	1.29E-07	6.82E-09												
ResNet	7.06E-12	2.05E-11	1.10E-09	2.66E-09	**2.49E-01**											
MLP	1.08E-14	5.94E-14	1.48E-13	2.96E-13	1.16E-09	4.17E-08										
MCNN	2.98E-12	1.22E-11	1.88E-11	1.00E-11	1.97E-09	2.76E-09	9.63E-14									
COTE	2.21E-09	1.53E-08	8.01E-07	7.49E-08	1.82E-02	**1.89E-01**	9.04E-09	1.77E-09								
PROP	1.04E-11	5.06E-11	1.64E-10	5.37E-11	3.21E-07	2.25E-06	1.18E-05	1.40E-11	5.29E-08							
BOSS	1.05E-11	1.43E-10	2.46E-09	2.68E-10	**1.59E-03**	**1.88E-03**	9.64E-04	3.92E-11	1.26E-06	**4.74E-03**						
BOSS-VS	1.42E-13	8.39E-13	1.09E-12	3.19E-12	6.00E-08	2.37E-07	**2.31E-02**	1.08E-12	7.53E-09	**1.46E-03**	8.36E-04					
TSBF	6.01E-13	3.82E-12	4.74E-13	2.71E-12	1.41E-07	1.80E-07	**4.60E-02**	3.49E-13	2.68E-09	2.73E-05	**1.17E-02**	**6.65E-01**				
1NN-DTW	5.54E-14	2.38E-13	7.31E-14	3.28E-13	4.84E-12	5.38E-12	**1.84E-01**	7.06E-14	1.18E-12	1.10E-08	2.09E-07	1.44E-04	7.37E-05			
1NN-BOSS	3.27E-11	2.87E-10	8.41E-08	7.96E-09	1.69E-02	**8.58E-02**	1.61E-06	2.27E-10	5.34E-04	3.22E-04	2.07E-02	1.77E-08	7.97E-05	2.58E-11		
DTW	1.06E-14	5.13E-14	1.96E-14	1.58E-13	2.66E-12	8.56E-13	2.08E-01	1.41E-14	5.52E-13	7.24E-10	2.80E-08	3.99E-04	1.68E-04	**4.49E-01**	1.36E-10	
ED	2.45E-15	6.18E-15	9.52E-15	2.42E-14	7.19E-13	2.23E-13	4.67E-07	2.36E-15	9.97E-14	5.96E-12	6.05E-11	3.63E-08	7.00E-08	9.80E-04	5.53E-13	**2.03E-03**

approaches and outperform on 30 and 18 datasets, respectively. These proposed models also secure the lowest arithmetic mean rank and MPCE score among all the methods. Figure 3 indicates the superiority of proposed models over the existing state-of-the-art methods in terms of MPCE score. The Wilcoxon signed-rank test also provides the statistical comparison of each model and shows the efficiency of the proposed models over other approaches, Table 5. The proposed models also depict better performance over other methods on the ShapesAll dataset, which has the highest number of classes (60 classes) among all the datasets in the UCR archive. Although ABiLSTM-FCN underperformed as compared to BiLSTM-FCN but also showed a slightly small difference in MPCE score, which is 0.03020 and 0.03060, respectively. The average arithmetic mean of BiLSTM-FCN and ABiLSTM-FCN is 3.0823 and 3.3529. The present existing state-of-the-art model, LSTM-FCN and ALSTM-FCN, has an average arithmetic mean of 4.4588 and 4.2705, respectively. Figure 4a, b shows the training and validation error loss of BiLSTM-FCN and ABiLSTM-FCN models on the FaceAll and ShapesAll datasets.

Figure 5 shows the critical difference diagram that is a pairwise statistical difference comparison among proposed models and other 15 state-of-the-art



BiLSTM-FCN

ABiLSTM-FCN

BiLSTM-FCN

ABiLSTM-FCN

Fig. 4 **a** The training and validation error of BiLSTM-FCN and ABiLSTM-FCN models over the *FaceAll* dataset, **b** The training and validation error of BiLSTM-FCN and ABiLSTM-FCN models over the *ShapesAll* dataset

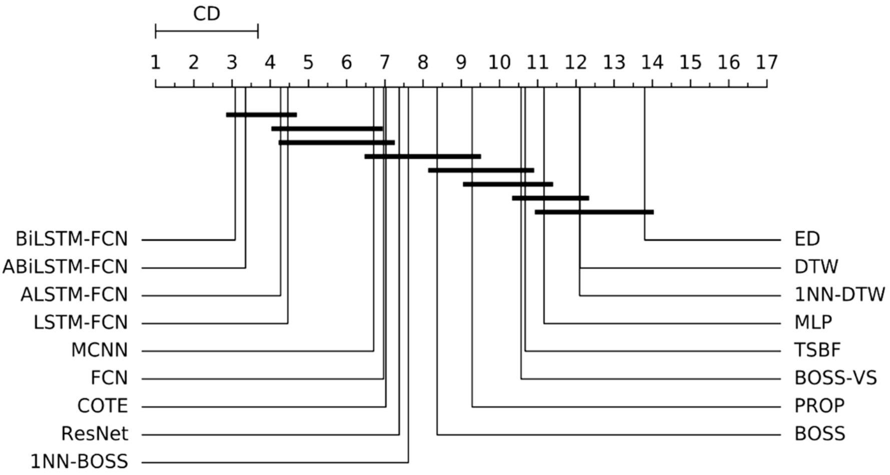


Fig. 5 Critical difference diagram showing pairwise statistical difference comparison between our proposed models and other 15 state-of-the art methods based on arithmetic mean rank

methods based on the arithmetic mean rank of testing error (lower is better). The visual representation shows that our models perform equally well compared with LSTM-FCN and ALSTM-FCN. FCN, COTE, and DTW, 1NN-DTW depict similar performance in the critical difference diagram with each other. Figure 6a, b shows the comparison of the proposed model and other well-known methods in terms of testing error rate on the different datasets.

Table 4 shows the performance comparison based on the f1-score. In terms of f1-score, BiLSTM-FCN and ABiLSTM-FCN depict superior performance over 38 and 32 datasets, respectively, while LSTM-FCN and ALSTM-FCN show better performance over 19 and 17 datasets, respectively.

Our main motivation to augment BiLSTM with FCN was to exploit input data from both the left and right directions to capture past and future hidden information, and FCN has already proven good at feature extraction, so by using FCN, there is no need for heavy feature engineering so both can outperform together as a hybrid deep neural network. The results demonstrate the success of this proposed architecture. Besides, the attention mechanism did not show better performance with BiLSTM as compared to LSTM. One of the limitations of BiLSTM is, it contains more parameters than unidirectional LSTM that can exceed training time, but this issue is resolved by using the dimension shuffle layer. Dimension shuffle helps to reduce the training time by using a single time step at the input stage of BiLSTM or AbiLSTM. Therefore, the proposed models are small enough to train and easy to deploy on real-time applications and for different classification tasks related to Image, Sensor, Motion, Spectro, Device, ElectroCardioGram (ECG), without any heavy data pre-processing, feature engineering and refinement; they are robust and efficient.

The experimental results can be summarized as follows:

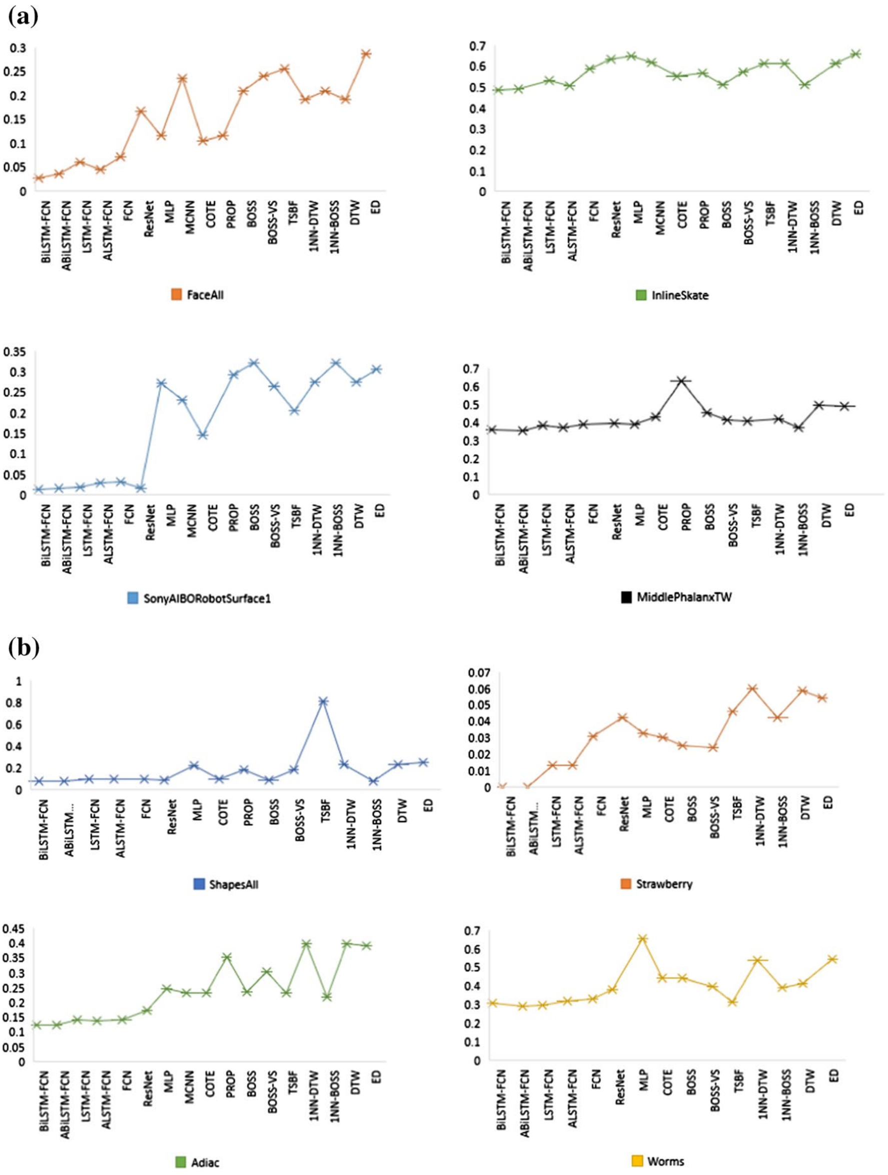


Fig.6 **a** Classification testing error on different datasets, **b** classification testing error rate on different datasets

- To evaluate the performance of our proposed models, we showed a comparison with the other existing best methods that claim the state-of-the-art results and baselines.
- The classification testing error loss and f1-score are used to evaluate the performance.

- After evaluating the performance over 85 datasets from different domains, using more than one evaluation metrics, it is evident that our proposed models produced better results than existing state-of-the-art techniques.

5 Conclusion

We have presented two hybrid deep learning models BiLSTM-FCN and ABiLSTM-FCN, for end-to-end univariate TSC. Extensive experiments on the UCR time series archive show that the proposed models produce results that are outperforming over some recognized methods and state-of-the-art techniques. The results also indicate that the BiLSTM-FCN model achieves superior performance over LSTM-FCN as LSTM tends to ignore future contextual information while processing data sequences in a time series. The proposed model performed better without any data pre-processing or refinement. The attention mechanism also strengthened the result by aggregating hidden states and naturally capturing the long-range temporal information of the inputs. Besides, we have further experimented our proposed models with the absence and presence of dropout to show its significance in neural networks. An overall comparison is demonstrated between proposed models and all the baselines and state-of-the-art techniques. The proposed models achieve better performance on several UCR archive datasets.

Acknowledgments This paper was partially supported by NSFC grant U1509216, U1866602, 61602129, and Microsoft Research Asia.

References

1. Esling P, Agon C (2012) Time-series data mining ACM computing surveys (CSUR) 45(1):12
2. Wei L, Keogh E (2006) Semi-supervised time series classification. In proceedings of the 12th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining
3. Karim F, Majumdar S, Darabi H (2019) Adversarial attacks on time series. https://doi.org/10.1109/TPAMI.2020.2986319
4. Chen Y, Keogh E, Hu B, Begum N, Bagnall A, Mueen A, Batista G (2015) The UCR time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data/
5. Dau HA et al. (2018) The ucr time series archive. https://doi.org/10.1109/JAS.2019.1911747
6. Keogh E, Ratanamahatana CA (2005) Exact indexing of dynamic time warping. Knowl Inf Syst 7(3):358–386
7. Rakthanmanon T et al. (2012) Searching and mining trillions of time series subsequences under dynamic time warping. Proceedings of the 18th ACM SIGKDD International Conference On Knowledge Discovery And Data Mining
8. Lin J et al. (2007) Experiencing SAX: a novel symbolic representation of time series. Data Min Knowl Disc 15(2):107–144
9. Baydogan MG, Runger G, Tuv E (2013) A bag-of-features framework to classify time series. IEEE Trans Pattern Anal Mach Intell 35(11):2796–2802
10. Schäfer P (2015) The boss is concerned with time series classification in the presence of noise. Data Min Knowl Disc 29(6):1505–1530
11. Schäfer P (2016) Scalable time series classification. Data Min Knowl Disc 30(5):1273–1298
12. Sch P et al. (2017) Fast and accurate time series classification with WEASEL, in Proceedings of the 2017 ACM on Conference On Information And Knowledge Management. ACM Singapore. 637–646
13. Lines J, Bagnall A (2015) Time series classification with ensembles of elastic distance measures. Data Min Knowl Disc 29(3):565–592

14. Bagnall A et al (2015) Time-series classification with COTE: the collective of transformation-based ensembles. IEEE Trans Knowl Data Eng 27(9):2522–2535
15. Wang Z, Yan W, Oates T (2017) Time series classification from scratch with deep neural networks: A strong baseline. In International Joint Conference On Neural Networks (IJCNN)
16. Cui Z, Chen W,Chen Y (2016) Multi-scale convolutional neural networks for time series classification. arXiv preprint https://arxiv.org/abs/1603.06995
17. Karim F et al. (2018) LSTM fully convolutional networks for time series classification. IEEE Access 6:1662–1669
18. Ortego P et al. (2020) Evolutionary LSTM-FCN networks for pattern classification in industrial processes. Swarm Evolut Comput 54:100650
19. Kim Y et al. (2018) Resource-efficient pet dog sound events classification using LSTM-FCN based on time-series data. Sensors 18(11):4019
20. Hashida S, Tamura K (2019) Multi-channel MHLF: LSTM-FCN using MACD-histogram with multichannel input for time series classification. in (2019) IEEE 11th International Workshop on Computational Intelligence and Applications (IWCIA)
21. Budak Ü et al. (2019) Computer-aided diagnosis system combining FCN and Bi-LSTM model for efficient breast cancer detection from histopathological images. Appl Soft Comput 85:105765
22. Jiang F, Chen H, Zhang L-J (2018) FCN-biLSTM based VAT invoice recognition and processing. IN INTERNATIONAL CONFERENCE ON EDGE COMPUTING, Springer
23. Srivastava N et al. (2014) Dropout: a simple way to prevent neural networks from overfitting. J Machine Learn Res 15(1):1929–1958
24. Ismail Fawaz H et al. (2019) Deep learning for time series classification: a review. Data Mining and Knowledge Discovery
25. Graves A, Schmidhuber J (2005) Framewise phoneme classification with bidirectional LSTM and other neural network architectures. Neur Networks 18(5–6):602–610
26. Schuster M, Paliwal KK (1997) Bidirectional recurrent neural networks. IEEE Trans Sign Proc 45(11):2673–2681
27. Lipton ZC, Berkowitz J, Elkan C (2015) A critical review of recurrent neural networks for sequence learning. arXiv preprint https://arxiv.org/abs/1506.00019
28. Hochreiter S, Schmidhuber J (1997) Long short-term memory. Neural Comput 9(8):1735–1780
29. Zhao Y et al. (2018) Applying deep bidirectional LSTM and mixture density network for basketball trajectory prediction. Optik 158:266–272
30. Graves A et al. (2009) A novel connectionist system for unconstrained handwriting recognition. IEEE Trans Pattern Anal Mach Intell 31(5):855–868
31. Chen T et al. (2017) Improving sentiment analysis via sentence type classification using BiLSTM-CRF and CNN. Expert Syst Appl 72:221–230
32. Graves A, Jaitly N, Mohamed Ar (2013) Hybrid speech recognition with deep bidirectional LSTM. in 2013 IEEE workshop on automatic speech recognition and understanding
33. Ioffe S, Szegedy C (2015) Batch normalization: accelerating deep network training by reducing internal covariate shift. arXiv preprint https://arxiv.org/abs/1502.03167
34. Nair V , Hinton GE (2010) Rectified linear units improve restricted boltzmann machines. In Proceedings of the 27th International Conference On Machine Learning (ICML-10)
35. Lin M, Chen Q, Yan S (2013) Network in network. arXiv preprint https://arxiv.org/abs/1312.4400
36. Bahdanau D, Cho K, Bengio Y (2014) Neural machine translation by jointly learning to align and translate. arXiv preprint https://arxiv.org/abs/1409.0473
37. Chorowski JK et al. (2015) Attention-based models for speech recognition. Adv Neural Info Process Systems. 28:577–585
38. Xu K et al. (2015) Show attend and tell: Neural image caption generation with visual attention. In International Conference On Machine Learning
39. Zhou Q, Wu H (2018) NLP at IEST 2018: BiLSTM-Attention and LSTM-Attention via Soft Voting in Emotion Classification. In Proceedings of the 9th Workshop on Computational Approaches to Subjectivity, Sentiment and Social Media Analysis
40. Tang Y et al. (2016) Sequence-to-sequence model with attention for time series classification. In 2016 IEEE 16th International Conference On Data Mining Workshops (ICDMW)
41. Vinayavekhin P et al. (2018) Focusing on what is relevant: time-series learning and understanding using attention. In 2018 24th International Conference On Pattern Recognition (ICPR)
42. Kingma DP, Ba Adam J (2014) A method for stochastic optimization. arXiv preprint https://arxiv.org/abs/1412.6980

43. Chollet F, Keras (2015) Available from: https://github.com/fchollet/keras
44. Abadi M et al. (2016) Tensorflow: a system for large-scale machine learning. In 12th {USENIX} Symposium on operating systems design and implementation ({OSDI} 16)
45. Nolan JR (1997) Estimating the true performance of classification-based nlp technology. In: From research to commercial applications: Making NLP Work in Practice
46. Powers DM (2020) Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation. arXiv preprint https://arxiv.org/abs/2010.16061

Publisher's Note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Affiliations

Mehak Khan[1] · Hongzhi Wang[1] · Adnan Riaz[2] · Aya Elfatyany[1] · Sajida Karim[1]

Hongzhi Wang
wangzh@hit.edu.cn

Adnan Riaz
adnanriaz107@mail.dlut.edu.cn

Aya Elfatyany
ayaelfatyany@hit.edu.cn

Sajida Karim
sajidakarim@hit.edu.cn

[1] School of Computer Science and Technology, Harbin Institute of Technology, Harbin 150001, China

[2] School of Computer Science and Technology, Dalian University of Technology, No. 2 Linggong Road, Ganjingzi District, Dalian 116024, China