information

Article

LA-ESN: A Novel Method for Time Series Classification

Hui Sheng, Min Liu, Jiyong Hu, Ping Li *, Yali Peng and Yugen Yi *

School of Software, Jiangxi Normal University, Nanchang 330022, China
* Correspondence: 003173@jxnu.edu.cn (P.L.); yiyg510@jxnu.edu.cn (Y.Y.)

Abstract: Time-series data is an appealing study topic in data mining and has a broad range of applications. Many approaches have been employed to handle time series classification (TSC) challenges with promising results, among which deep neural network methods have become mainstream. Echo State Networks (ESN) and Convolutional Neural Networks (CNN) are commonly utilized as deep neural network methods in TSC research. However, ESN and CNN can only extract local dependencies relations of time series, resulting in long-term temporal data dependence needing to be more challenging to capture. As a result, an encoder and decoder architecture named LA-ESN is proposed for TSC tasks. In LA-ESN, the encoder is composed of ESN, which is utilized to obtain the time series matrix representation. Meanwhile, the decoder consists of a one-dimensional CNN (1D CNN), a Long Short-Term Memory network (LSTM) and an Attention Mechanism (AM), which can extract local information and global dependencies from the representation. Finally, many comparative experimental studies were conducted on 128 univariate datasets from different domains, and three evaluation metrics including classification accuracy, mean error and mean rank were exploited to evaluate the performance. In comparison to other approaches, LA-ESN produced good results.

Keywords: time series classification; echo state network; long short-term memory network; convolutional neural network

check for updates

Citation: Sheng, H.; Liu, M.; Hu, J.; Li, P.; Peng, Y.; Yi, Y. LA-ESN: A Novel Method for Time Series Classification. *Information* **2023**, *14*, 67. https://doi.org/10.3390/info14020067

Academic Editors: María N. Moreno García and Francesco Camastra

Received: 9 November 2022
Revised: 14 January 2023
Accepted: 19 January 2023
Published: 26 January 2023

1. Introduction

Massive data resources have accumulated in numerous industries in the quickly increasing information era, and large-scale data provide valuable content. Time series data is a set of data points arranged in chronological order, which can be divided into two categories: univariate and multivariate. In univariate time series data, only one variable varies over time, while in multivariate time series data, multiple variables change over time [1]. Time series data are now widely used in a wide range of applications, including statistics, pattern recognition, earthquake prediction, econometrics, astronomy, signal processing, control engineering, communication engineering, finance, weather forecasting, the Internet of Things, and medical care [2].

Much work has been completed around the TSC problem in the last two decades. This enlightened and attractive data mining topic focuses on classifying data points indexed by time by predicting their labels. Time series classification is an essential task in data mining and has been extensively employed in many fields. For example, in the field of network traffic, Xiao et al. [3] proposed traffic classification for dynamic network flows. In medical diagnosis, R. Michida et al. [4] proposed a deep learning method for classifying lesions based on JNET classification for computer-aided diagnosis systems for colorectal magnification NBI endoscopy. In finance, Mori et al. [5] and Wan et al. [6] proposed methods for the early classification of time series using multi-objective optimization techniques and financial strategies for classifying chart patterns in time series.

Generally, the solution to the TSC task can be divided into three types [7]: (1) Distance-based approaches: the distances between various samples are evaluated and utilized for classification through a distance function. Representative distance-based techniques include Time-warping Edit Distance (TED) [8], Weighted Dynamic Time Warping (WDTW) [9], Complexity Invariant Distance (CID) [10], Derivative Transform Distance (DTD) [11], Dynamic Derivative Time Warping (DDTW) [12], 1-Nearest Neighbour with Euclidean Distance (ED) [13] and Longest Common Subsequence (LCSS) [14]. (2) Feature-based methods: use feature extraction to extract compelling features from the original data, which can be local features of the data or global features. Examples include Time Series Bag Feature (TSBF) [15], Time Series Forest (TSF) [16], Fast Shape Tree (FS) [17], Shape Transform (ST) [18], Learning Pattern Similarity (LPS) [19], and Dynamic Time Warping Feature (DTWF) [20]. (3) Neural network-based methods: classifying time series through an "end-to-end" learning model. The raw data are fed straight into the model, and the model generates direct results. For example, Long Short-Term Memory Fully Convolutional Network (LSTM-FCN) [21], Echo State Network (ESN) [22], Attention LSTM-FCN (ALSTM-FCN) [23], and Temporal Convolutional Network (TCN) [24].

ESN, CNN and LSTM are widely used for time series classification tasks. Using ESNs alone is insufficient for time series classification. Therefore, several researchers have proposed models that fuse ESNs with CNNs to make ESNs more adaptable to time series classification tasks. Ma et al. [22] used ESN to generate a time series representation matrix before employing CNN to extract classification features. LSTM is widely used to extract long-term dependencies in time series, and some researchers have also fused CNN with LSTM for time series classification and achieved quite good results. For example, Karim et al. [21] incorporated a full convolutional block and an LSTM to propose a new method. The full convolutional block contains three temporal convolutional blocks as feature extractors, and the temporal convolution block has a convolutional layer with multiple filters. However, extracting features directly from the time series for classification necessitates substantial preprocessing. The attention mechanism has become an essential concept in deep learning and has been successfully applied to time series tasks. In recent years, the main principle behind employing attention in time series classification has been to focus on the most important information related to the input data while extracting features from the input data.

In light of the preceding work, we propose an end-to-end TSC model called LA-ESN for time series classification. The LA-ESN consists of an echo memory encoder and a decoder. The reservoir layer makes up the echo memory encoder, while the decoder is made up of a 1D CNN, an LSTM and Attention. The storage layer first projects the time series into a high-dimensional nonlinear space to generate echo states step by step. The echo memory matrix is formed by collecting the echo states of all time steps in chronological order. To capture the critical historical information in the time series, we design a decoder that applies a 1D CNN, an LSTM and Attention to the echo memory matrix, respectively. To increase network efficiency, the 1D CNN and LSTM are employed to extract multi-scale features and retrieves global information from the echo memory matrix. Then, the Attention is used to extract the essential information from the global information. LA-ESN is an acceptable classification approach, according to experimental findings on a variety of time series datasets. The following are the main contributions of this paper:

(1) We propose a simple end-to-end model LA-ESN for handling time series classification tasks;
(2) We modify the output layer of ESN to handle time series better and use CNN and LSTM as output layers to finish feature extraction;
(3) The attention mechanism is deployed behind both CNN and LSTM, which effectively improves the effectiveness and computing efficiency of LA-ESN;
(4) Experiments on various time series datasets show that LA-ESN is efficacious.

The rest of this paper is organized as follows. Section 2 discusses the related work of this study. Section 3 explains the proposed method. Section 4 provides a description of the UCR time series database and the results of the comparison experiments. A brief conclusion is given in Section 5.

2. Associated Work

2.1. CNN with Attention

CNN and AM have been broadly adopted in image processing and other fields. In combination with AM, CNN has been proposed for time series classification. AM can be interpreted as a feature enhancement method by extracting the most information-rich components of a signal. After calculating the attention values based on the saliency importance in the feature maps extracted from the convolutional layers, the refined feature map weights can be derived from the attention values. A Deep CNN (DCNN) with an attention module has been proposed as a framework for time series classification [25]. This network improves the classification performance for various seismic events and solves all possible seismic events. Tripathi et al. [26] proposed an Attention-based Multivariate CNN (AT-MVCNN), which contains an input tensor based on attention features to encode information across multiple timestamps. Sun et al. [27] proposed a Prototypical Inception Network with Cross Branch Attention (PIN-BA) framework that uses CNNs with various receiver window branches to capture features at different time window scales, using a cross-branching attention scheme to emphasize critical feature information in the classification process. In [28], five AMs were applied to six neural networks to focus on the valid information in the time series from either the channel or the spatial dimension. All of the above studies yielded promising results in time series classification.

2.2. ESN -Based Classifier

An ESN is a recursive structure consisting of random RNNs. Although ESNs are mainly employed to predict time series, many ESN-based classification networks have also been designed to implement the TSC task. For example, using Autoencoder (AE) theory, Wang et al. [7] developed an ESN with an input weight framework and a globally reversible algorithm to reconstruct the randomly initialized input weights of the ESN network. Existing self-encoder and ESN-based network models start with initial input weights, the output weights generated during the encoding process. Huang et al. [29] proposed an innovative ESN approach named Functional Deep Echo State Network (FDESN), which introduced temporal and spatial aggregation. Moreover, this approach considers the relative value of temporal data throughout several periods and the dynamic properties of MTS. Wang et al. [30] offered a unique Discriminative Regularized ESN (DR-ESN) time series classification algorithm by combining Discriminative Feature Aggregation (DFA) and an Outlier Robust Weight (ORW) algorithm. First, the DFA algorithm is adopted to replace the random input weights of the ESN with the bounded weights based on the sample information. Second, the ORW algorithm is applied to weight-constrained samples with notable training errors to achieve greater robustness in the training process. The DR-ESN can effectively improve the classification performance of the original ESN and significantly reduce the impact of outliers on the classification result.

3. The Proposed LA-ESN Framework

The framework of LA-ESN is divided into two parts: encoding and decoding. In the encoding part, each frame of the input time series is mapped into a high-dimensional reservoir state space to obtain an Echo State Representation (ESR). The ESRs of all time steps are stored in a memory matrix. We simultaneously design two ways to decode the memory matrix in the decoding phase. First, we adopt a Multi-scale 1D CNN [31–33] to extract the local information. Second, utilizing LSTM and Attention, long-term dependent information is recovered from the memory matrix. Both attention mechanisms are intended to reduce irrelevant information while increasing computational efficiency. Finally, the local and long-term dependent information obtained by the two methods is pooled and merged, and then the merged features are passed through a fully connected layer. The conditional probability distribution of the categories is calculated using a Softmax layer. The proposed LA-ESN model's general design is seen in Figure 1.

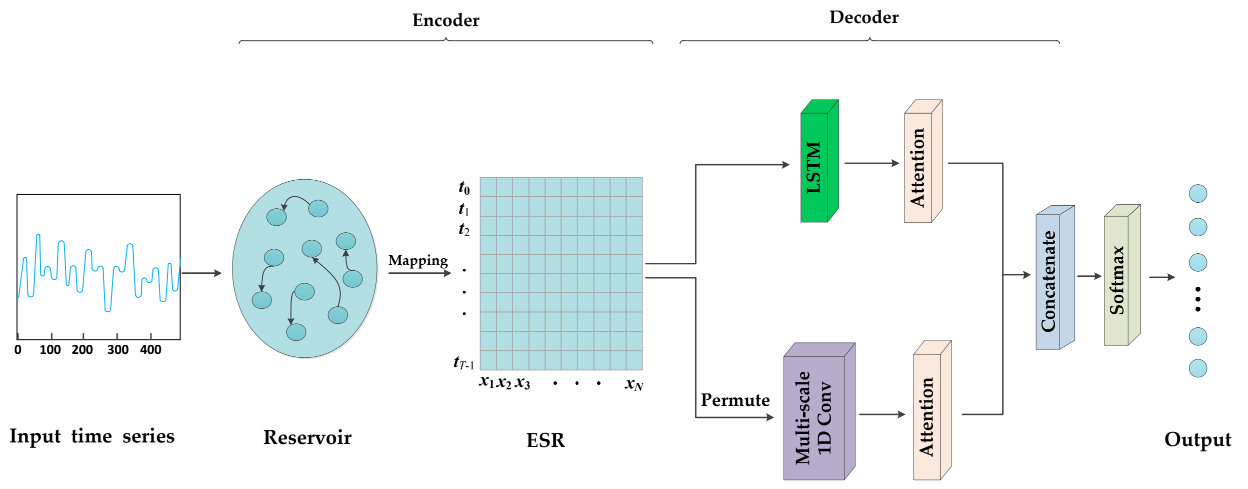

Figure 1. The general architecture of LA-ESN model.

3.1. Preliminary

TSC refers to the identification of non-labeled samples based on the given labeled samples. The time series dataset, consisting of N samples, can be expressed as follows:

$$\begin{gathered} D = \{(X_1, y_1), \cdots, (X_i, y_i), \cdots, (X_N, y_N)\} \\ X_i = (X_i(1), \cdots, X_i(t), \cdots, X_i(T)) \end{gathered} \tag{1}$$

where T denotes the time stamp of the time series data and y_i represents the corresponding category in each time series X_i.

3.2. Encoding Stage

The primitive ESN includes an input, reserve pool, and output layer. The diagram of the primitive ESN model is shown in Figure 2. The primitive ESN uses a reserve pool of randomly sparsely connected neurons as the hidden layer of a high-dimensional and nonlinear input representation. The weights of the hidden layer of the ESN are generated in advance rather than by training. Meanwhile, they are trained from the hidden layer to the output layer. Therefore, the generated reserve pool has some good properties that guarantee excellent performance by using only linear methods to train the weights from the reserve pool to the output layer. Given a k-dimensional input, $i(t)$ with time step t, the state of the reserve pool with time step $t-1$ is $r(t-1)$. The equations for updating the primitive ESN are as follows:

$$r(t) = f\Big(w^{res} r\Big(t-1) + w^{in} i(t)\Big) \tag{2}$$

$$o(t) = f^{out}\big(w^{out} r(t)\big) \tag{3}$$

$$w^{res} = SR \times \frac{w}{\lambda_{max}(w)} \tag{4}$$

where w^{in},w^{res} and w^{out} denote the input, intermediate and output weight matrix, respectively; $i(t)$ and $r(t-1)$ express the time series input and the reservoir state at step $t-1$, respectively; w^{in} and w^{out} are randomly initialized and fixed; w^{res} can be calculated by Equation (4), SR is the spectral radius of w^{res}, $\lambda_{max}(w)$ is the maximum eigenvalue of the matrix W and the elements of W are generated randomly in $[-0.5, 0.5]$.

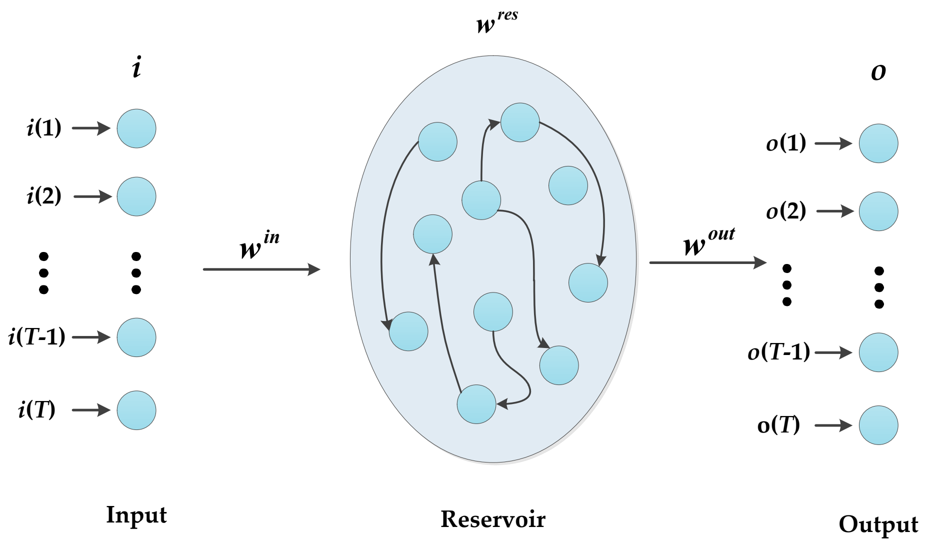

Figure 2. The diagram of the primitive ESN model.

In this work, we take ESN as the base network and modify the output layer of ESN to make the network model more suitable for TSC, as shown in Figure 1. Suppose $s = (s(0),\ s(1),\ \cdots,\ s(T-1))^T$ is a k-dimensional time series. At each time step, the echo state $r(t)$ is calculated according to Equation (2) and the echo state memory matrix R can be obtained as follows:

$$R = \begin{pmatrix} r_1(0) & r_2(0) & \cdots & r_N(0) \\ r_1(1) & r_2(1) & \cdots & r_N(1) \\ \vdots & \vdots & \vdots & \vdots \\ r_1(T-1) & r_2(T-1) & \cdots & r_N(T-1) \end{pmatrix} \tag{5}$$

We use $r(t)$ to indicate the t-th row of R, which denotes the echo state at the t-th time step where $t \in 0,\ 1,\ \ldots,\ (T-1)$. The term r_m demonstrates as the m-th column of R, expressing the echo state of the m-th time sequence at all time steps where $m \in 0,\ 1, \ldots,\ (T-1)$. We keep all the calculated echo states in R to obtain a complete ESR of the sequence as the input of the decoder, and the decoder extracts the discriminative features to determine the category labels.

3.3. Decoding Stage

In previous studies, multi-scale convolution has been used as a feature extractor to extract effective classification features from time series representations. Alternatively, LSTM is used to learn straightforwardly from the input time series. Both approaches are capable of classifying time series to some extent, although they might be improved. Therefore, we propose to appropriately adapt and then combine them to be used as the output layer of ESN to learn better feature information from the echo states.

On the one hand, we employ multiple scales of 1D CNNs for convolution operations along the time direction and use multiple filters for each time scale. The batch normalization operation follows the convolution operations to avoid the gradient disappearance problem. Next, ReLU as a correction layer for the activation function is adopted, which can improve the nonlinear and sparsity connection between the levels to reduce over-fitting. Therefore, batch normalization and ReLU can achieve more robust learning. Finally, the multi-scale features are concatenated into a new feature. The structure diagram of a multi-scale 1D convolution is shown in Figure 3.

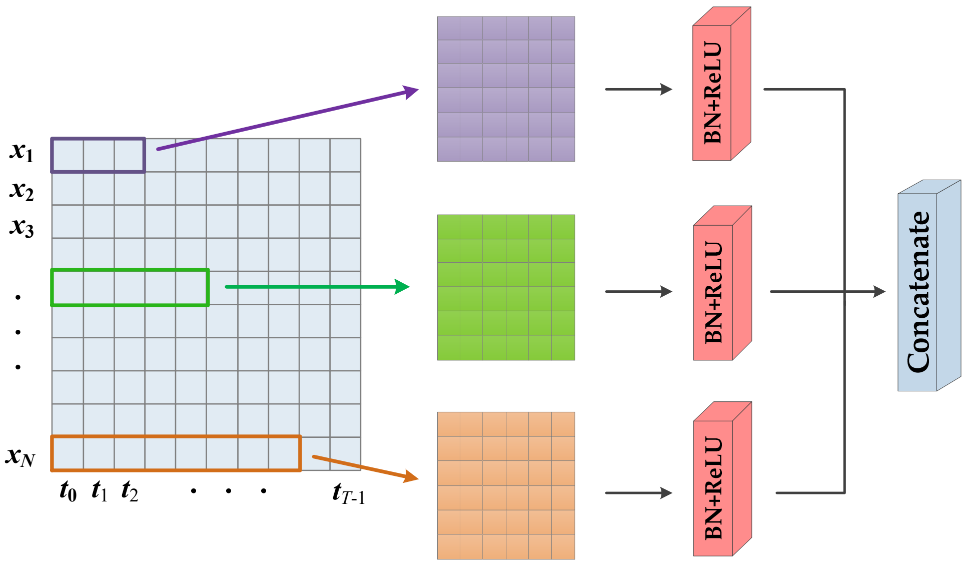

Figure 3. The diagram of a multi-scale 1D convolution. The stride of each 1D convolution in LA-ESN is set to 1. Different colours represent the convolution results with different convolution kernel sizes. From top to bottom, the convolution results are shown for convolution kernel sizes 3, 5 and 8, respectively.

On the other hand, LSTM is extremely successful at dealing with time series challenges. Therefore, we propose to learn the long-term global dependence between the states from the echo state matrix using LSTM to make LA-ESN learn more robust classification feature information. In LA-ESN, the LSTM receives the echo state matrix as a multivariate state matrix with a single time step, and the operations of the LSTM are shown as follows:

$$f_t = \sigma\left(W_f h_{t-1} + W_f x_t + b_f\right) \tag{6}$$

$$i_t = \sigma(W_i h_{t-1} + W_i x_t + b_i) \tag{7}$$

$$o_t = \sigma(W_o h_{t-1} + W_o x_t + b_o) \tag{8}$$

$$w_t = \tan\mathrm{h}(W_w h_{t-1} + W_w x_t + b_w) \tag{9}$$

$$C_t = f_t \otimes C_{t-1} + i_t \otimes w_t \tag{10}$$

$$h_t = o_t \otimes \tan\mathrm{h}(C_t) \tag{11}$$

where σ defines as the logistic sigmoid function, and symbol $\otimes$ represents the element-wise multiplication. The recurrent weight matrices are depicted using the notations W_f, W_i, W_o, and W_w. The deviation is depicted using the notation b_f, b_i, b_o and b_w. The diagram of the LSTM model is shown in Figure 4.

Then, the attention module is integrated, which is commonly used in natural language processing. The context vector c_i depends on a sequence of annotations $(h_i, \cdots, h_{T_x})$. Each annotation h_i contains information about the entire input sequence, focusing on the part of the input sequence around the i-th word. The encoder maps the input sequence to a new sequence. The context vector c_i is a weighted sum of these annotations as below:

$$c_i = \sum_{j-1}^{T_x} a_{ij} h_j \tag{12}$$

The weight a_{ij} for each annotation h_j is calculated as follows:

$$a_{ij} = \frac{\exp(e_{ij})}{\sum_{k\text{-}1}^{T_x} \exp(e_{ik})} \tag{13}$$

$$e_{ij} = \varphi(s_{i-1}, h_j) \tag{14}$$

where h_j denotes the j-th hidden state vector of the encoder and s_{i-1} denotes the hidden state of the decoder at the previous time step. The term φ is a scoring function and used to calculate the similarity between h_j and s_{i-1}. Figure 5 depicts the attention model diagram. There are two advantages to using the AM. First, AM can apply various weights to distinct echo state representations of the same time step. In other words, AM can give more weight to information that is more significant for categorization while suppressing irrelevant data. Second, including the AM increases the model's running speed.

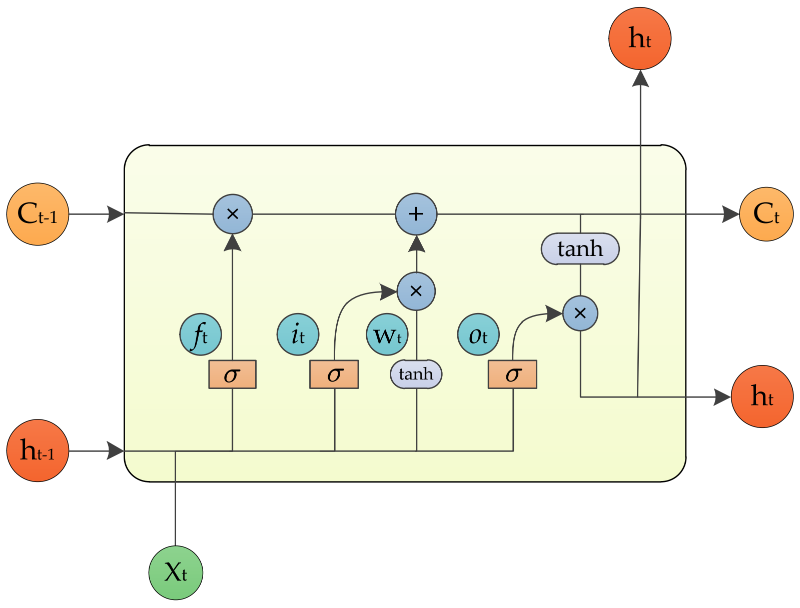

Figure 4. The diagram of LSTM model.

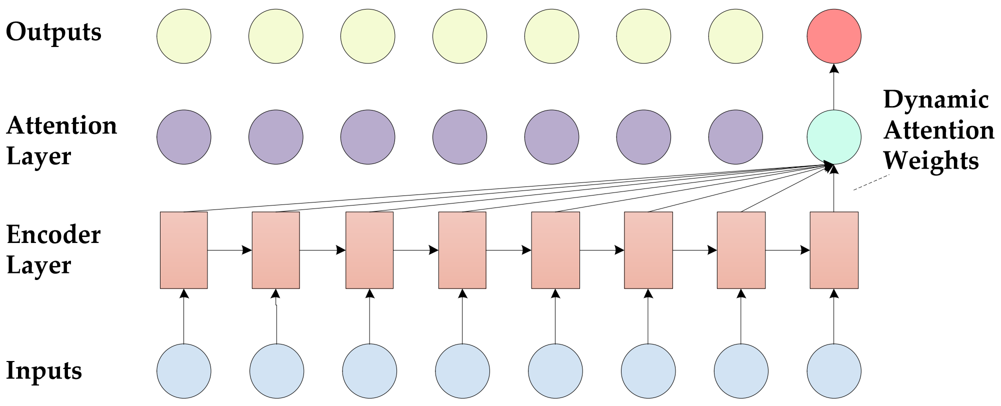

Figure 5. The diagram of attention model.

4. Experiments and Results

4.1. Database Description

We conducted extensive tests using the publicly accessible UCR database, which can be acquired from this URL (https://www.cs.ucr.edu/~eamonn/time_series_data_2018/ (accessed on 9 October 2022)), to ensure that the proposed LA-ESN approach is valid for the time series classification problem. The collection comprises 128 datasets divided into 15 categories. The specific descriptions of the datasets used are listed in Table 1. Each dataset has five characteristics: training set size, test set size, number of categories, sequence length and category [34]. The selected datasets cover multiple categories in the UCR database. We use Python programming language to implement our proposed LA-SEN approach and execute all tests on an Intel Core i9-9900 k CPU with 32 GB RAM and an Nvidia GeForce RTX 2080 GPU.

Table 1. The specific description of 128 datasets used for time series classification.

Dataset	Train	Test	Class	Length
ACSF1	100	100	10	1460
Adiac	390	391	37	176
AllGestureX	300	700	10	0
AllGestureY	300	700	10	0
AllGestureZ	300	700	10	0
ArrowHead	36	175	3	251
Beef	30	30	5	470
BeetleFly	20	20	2	512
BirdChicken	20	20	2	512
BME	30	150	3	128
Car	60	60	4	577
CBF	30	900	3	128
Chinatown	20	345	2	24
ChlorineCon	467	3840	3	166
CinCECGTorso	40	1380	4	1639
Coffee	28	28	2	286
Computers	250	250	2	720
CricketX	390	390	12	300
CricketY	390	390	12	300
CricketZ	390	390	12	300
Crop	7200	16,800	24	46
DiatomSizeR	16	306	4	345
DistPhxAgeGp	400	139	3	80
DistlPhxOutCorr	600	276	2	80
DistPhxTW	400	139	6	80
DodgerLoopDay	78	80	7	288
DodgerLoopGame	20	138	2	288
DodgerLoopWnd	20	138	2	288
Earthquakes	322	139	2	512
ECG200	100	100	2	96
ECG5000	500	4500	5	140
ECGFiveDays	23	861	2	136
ElectricDevices	8926	7711	7	96
EOGHorSignal	362	362	12	1250
EOGVerticalSignal	362	362	12	1250
EthanolLevel	504	500	4	1751
FaceAll	560	1690	14	131
FaceFour	24	88	4	350
FacesUCR	200	2050	14	131
FiftyWords	450	455	50	270
Fish	175	175	7	463
FordA	3601	1320	2	500
FordB	3636	810	2	500

Table 1. *Cont.*

Dataset	Train	Test	Class	Length
FreezerRegularT	150	2850	2	301
FreezerSmallTrain	28	2850	2	301
Fungi	18	186	18	201
GestureMidAirD1	208	130	26	360
GestureMidAirD2	208	130	26	360
GestureMidAirD3	208	130	26	360
GesturePebbleZ1	132	172	6	0
GesturePebbleZ2	146	158	6	0
GunPoint	50	150	2	150
GunPointAgeSpan	135	316	2	150
GunPointMaleFe	135	316	2	150
GunPointOldYg	135	316	2	150
Ham	109	105	2	431
HandOutlines	1000	370	2	2709
Haptics	155	308	5	1092
Herring	64	64	2	512
HouseTwenty	34	101	2	3000
InlineSkate	100	550	7	1882
InsectEPGRegTra	62	249	3	601
InsectEPGSmallTra	17	249	3	601
InsectWingSnd	30,000	20,000	10	30
ItalyPowerDemand	67	1029	2	24
LargeKitchenApp	375	375	3	720
Lightning2	60	61	2	637
Lightning7	70	73	7	319
Mallat	55	2345	8	1024
Meat	60	60	3	448
MedicalImages	381	760	10	99
MelbournePed	1194	2439	10	24
MidPhxOutAgeGp	400	154	3	80
MidPhxOutCorr	600	291	2	80
MidPhxTW	399	154	6	80
MixedRegularTrain	500	2425	5	1024
MixedSmallTrain	100	2425	5	1024
MoteStrain	20	1252	2	84
NonFetalECGTh1	1800	1965	42	750
NonFetalECGTh2	1800	1965	42	750
OliveOil	30	30	4	570
OSULeaf	200	242	6	427
PhaOutCorr	1800	858	2	80
Phoneme	214	1896	39	1024
PickupGestureWZ	50	50	10	0
PigAirwayPressure	104	208	52	2000
PigArtPressure	104	208	52	2000
PigCVP	104	208	52	2000
PLAID	537	537	11	0
Plane	105	105	7	144
PowerCons	180	180	2	144
ProxPhxOutAgeGp	400	205	3	80
ProxPhaxOutCorr	600	291	2	80
ProxPhxTW	400	205	6	80
RefDevices	375	375	3	720
Rock	20	50	4	2844
ScreenType	375	375	3	720
SemgHandGenCh2	300	600	2	1500
SemgHandMovCh2	450	450	6	1500
SemgHandSubCh2	450	450	5	1500
ShakeGestureWZ	50	50	10	0

Table 1. *Cont.*

Dataset	Train	Test	Class	Length
ShapeletSim	20	180	2	500
ShapesAll	600	600	60	512
SmallKitchenApp	375	375	3	720
SmoothSubspace	150	150	3	15
SonyAIBORobSur1	20	601	2	70
SonyAIBORobSur2	27	953	2	65
StarLightCurves	1000	8236	3	1024
Strawberry	613	370	2	235
SwedishLeaf	500	625	15	128
Symbols	25	995	6	398
SyntheticControl	300	300	6	60
ToeSegmentation1	40	228	2	277
ToeSegmentation2	36	130	2	343
Trace	100	100	4	275
TwoLeadECG	23	1139	2	82
TwoPatterns	1000	4000	4	128
UMD	36	144	3	150
UWaveAll	896	3582	8	945
UwaveX	896	3582	8	315
UwaveY	896	3582	8	315
UwaveZ	896	3582	8	315
Wafer	1000	6164	2	152
Wine	57	54	2	234
WordSynonyms	267	638	25	270
Worms	181	77	5	900
WormsTwoClass	181	77	2	900
Yoga	300	3000	2	426

4.2. Evaluation Metric

In our experiment, the standard *accuracy* is used as the evaluation index. Meanwhile, the *mean error* (*ME*) and *mean rank* (*MR*) are employed to evaluate the classification performance of a given model on multiple datasets [35]. They are defined as follows:

$$PCE_k = \frac{e_k}{c_k} \tag{15}$$

$$ME = \frac{1}{K}\sum PCE_k \tag{16}$$

$$MR = \frac{SCR}{K} \tag{17}$$

where e_k denotes the error rate of the k-th dataset, c_k means the number of classes in the k-th dataset, and K represents the number of used datasets.

After sorting and numbering the data from largest to smallest, the *MR* is the average of the ordinal numbers in a dataset. The *MR* can help us see which of the methods is working well. The smaller average of *MR* may indicate that the model is more accurate than the other methods on most datasets.

To better understand *MR*, we introduce the *SCR* term with an example to explain the process of calculating the average rank. *SCR* represents the sorted sum of a particular method for all datasets. For example, we have three datasets and three methods denoted as $D = \{d_1, d_2, d_3\}$ and $F = \{f_1, f_2, f_3\}$, respectively, and the corresponding results of each method on each dataset are as follows:

$$\begin{array}{cccc} & f_1 & f_2 & f_3 \\ d_1 & 0.80 & 0.75 & 0.83 \\ d_2 & 0.79 & 0.87 & 0.89 \\ d_3 & 0.90 & 0.89 & 0.93 \end{array}$$

Then, according to accuracy on the dataset d_1, the f_1, f_2 and f_3 are sorted from large to small and denoted as {2, 3, 1}. In the same way, we obtain the sorting results on the datasets d_2 and d_3 are {3, 2, 1} and {2, 3, 1}, respectively. Thus, the rankings of f_1 are 2, 3 and 2 on dataset D, respectively, then the *SCR* of f_1 is the sum of sorts on all datasets; that is, the *SCR* of method f_1 is 7 = 2 + 3 + 2. According to Equation (17), the smaller *SCR*, the smaller *MR*. The method's accuracy is higher, and the *MR* on the dataset is more minor. Therefore, the *MR* can evaluate the overall classification results of the method on all datasets.

4.3. Results and Discussion

4.3.1. Compared with Traditional Methods

We compare the proposed LA-ESN model with the traditional TSC models to conduct a general evaluation. In this work, we selected 12 conventional models as comparison methods and abbreviated them as ED [12], DDTW [13], DTD [11], LS [14], BOSS [36], EE [37], FCOTE [38], TSF [16], TSBF [15], ST [18], LPS [19], and FS [17]. We briefly describe each comparison method as follows:

(1) ED (1-Nearest Neighbour with Euclidean Distance). In this method, the Euclidean distance is employed to measure the similarity of two given time series, and then the nearest neighbor is used for classification.
(2) DDTW (Dynamic Derivative Time Warping). This method weights the DTW distance between the two-time series and the DTW distance between the corresponding first-order difference series.
(3) DTD (Derivative Transform Distance). The DTW distance between sequences of sine, cosine, and Hilbert transforms is further considered based on the DDTW.
(4) LCSS (Longest Common Subsequence). In this method, a shapelet transform-based classifier is designed using a heuristic gradient descent-based shapelet search process instead of enumeration.
(5) BOSS (Bag of SFA Symbols). This method uses windows to form 'words' on the level and then explores a truncated discrete Fourier transform on each window to obtain features.
(6) EE (Elastic Ensemble). The EE method takes a voting scheme to combine eleven 1-NN classifiers with elastic distance metrics.
(7) FCOTE (Flat Collective of Transform-based Ensembles). The method integrates 35 classifiers by using the cross-validation accuracy of training sets.
(8) TSF (Time Series Forest). The time series is first divided into intervals in this method to calculate the mean, standard deviation and slope as interval features. Then, the intervals are randomly selected to train the tree forest.
(9) TSBF (Time Series Bag Feature). This method selects multiple random length subsequences from random locations and then divides these subsequences into shorter intervals to capture local information.
(10) ST (Shape Transform). This method uses the shapelet transform to obtain a new representation of the original time series. Then, a classifier is constructed on the new representation using a weighted integration of eight different classifiers.
(11) LPS (Learning Pattern Similarity). The method is based on intervals, but the main difference is that the subsequence is used as an attribute rather than the extracted interval features.
(12) FS (Fast Shape Tree). The method speeds up the search process by converting the original time series into a discrete low-dimensional representation by applying a

symbolic aggregation approximation to the actual time series. Random projections are then used to find potential shapelet candidates.

The average accuracy and standard deviation of our proposed LA-ESN method five times and other conventional TSC classifiers on 76 datasets are shown in Table 2. As seen from Table 2, we can conclude that the following: (1) LA-ESN achieves higher classification accuracy than the other 12 traditional methods on 36 datasets and achieves comparable results on the other 25 datasets. The total number significantly outperforms other methods. (2) The average classification accuracy of all datasets of the proposed LA-ESN is also superior to the other compared methods. (3) For *MR*, the performance of LA-ESN is only slightly worse than that of the FCOTE method but significantly better than that of the other methods. (4) The *ME* of the proposed LA-ESN is slightly higher than that of the FCOTE and ST methods but lower than that of the other methods. Therefore, the experimental results indicate that LA-ESN is effective for time series classification in most cases.

Table 2. The results of LA-ESN model and different traditional TSC modes on 76 datasets.

Dataset	ED	DDTW	DTD	LCSS	BOSS	EE	FCOTE	TSF	TSBF	ST	LPS	FS	LA-ESN
Adiac	0.611	0.701	0.701	0.522	0.765	0.665	0.790	0.731	0.770	0.783	0.770	0.593	0.984 (0.002)
Beef	0.667	0.667	0.667	0.867	0.800	0.633	0.867	0.767	0.567	0.900	0.600	0.567	0.935 (0.009)
BeetleFly	0.750	0.650	0.650	0.800	0.900	0.750	0.800	0.750	0.800	0.900	0.800	0.700	0.783 (0.080)
BirdChicken	0.550	0.850	0.800	0.800	0.950	0.800	0.900	0.800	0.900	0.800	1.000	0.750	0.775 (0.048)
Car	0.733	0.800	0.783	0.767	0.833	0.833	0.900	0.767	0.783	0.917	0.850	0.750	0.931 (0.002)
CBF	0.852	0.997	0.980	0.991	0.998	0.998	0.996	0.994	0.988	0.974	0.999	0.940	0.901 (0.030)
ChlorineCon	0.650	0.708	0.713	0.592	0.661	0.656	0.727	0.720	0.692	0.700	0.608	0.546	0.911 (0.009)
CinCECGTorso	0.897	0.725	0.852	0.869	0.900	0.942	0.995	0.983	0.712	0.954	0.736	0.859	0.909 (0.004)
Computers	0.576	0.716	0.716	0.584	0.756	0.708	0.740	0.720	0.756	0.736	0.680	0.500	0.551 (0.012)
CricketX	0.577	0.754	0.754	0.741	0.736	0.813	0.808	0.664	0.705	0.772	0.697	0.485	0.930 (0.004)
CricketY	0.567	0.777	0.774	0.718	0.754	0.805	0.826	0.672	0.736	0.779	0.767	0.531	0.933 (0.002)
CricketZ	0.587	0.774	0.774	0.741	0.746	0.782	0.815	0.672	0.715	0.787	0.754	0.464	0.935 (0.003)
DiatomSizeR	0.935	0.967	0.915	0.980	0.931	0.944	0.928	0.931	0.899	0.925	0.905	0.866	0.978 (0.005)
DistPhxAgeGp	0.626	0.705	0.662	0.719	0.748	0.691	0.748	0.748	0.712	0.770	0.669	0.655	0.803 (0.007)
DistPhxCorr	0.717	0.732	0.725	0.779	0.728	0.728	0.761	0.772	0.783	0.775	0.721	0.750	0.748 (0.009)
DistPhxTW	0.633	0.612	0.576	0.626	0.676	0.647	0.698	0.669	0.676	0.662	0.568	0.626	0.893 (0.005)
Earthquakes	0.712	0.705	0.705	0.741	0.748	0.741	0.748	0.748	0.748	0.741	0.640	0.705	0.734 (0.014)
ECG200	0.880	0.830	0.840	0.880	0.870	0.880	0.880	0.870	0.840	0.830	0.860	0.810	0.904 (0.010)
ECG5000	0.925	0.924	0.924	0.932	0.941	0.939	0.946	0.939	0.940	0.944	0.917	0.923	0.973 (0.000)
ECGFiveDays	0.797	0.769	0.822	1.000	0.983	0.820	0.999	0.956	0.877	0.984	0.879	0.998	0.883 (0.040)
ElectricDevices	0.552	0.592	0.594	0.587	0.799	0.663	0.713	0.693	0.703	0.747	0.681	0.579	0.895 (0.003)
FaceAll	0.714	0.902	0.899	0.749	0.782	0.849	0.918	0.751	0.744	0.779	0.767	0.626	0.974 (0.002)
FaceFour	0.784	0.829	0.818	0.966	0.996	0.909	0.898	0.932	1.000	0.852	0.943	0.909	0.910 (0.011)
FacesUCR	0.769	0.904	0.908	0.939	0.957	0.945	0.942	0.883	0.867	0.906	0.926	0.706	0.977 (0.003)
FiftyWords	0.631	0.754	0.754	0.730	0.705	0.820	0.798	0.741	0.758	0.705	0.818	0.481	0.988 (0.001)
Fish	0.783	0.943	0.926	0.960	0.989	0.966	0.983	0.794	0.834	0.989	0.943	0.783	0.965 (0.002)
FordA	0.665	0.723	0.765	0.957	0.930	0.738	0.957	0.815	0.850	0.971	0.873	0.787	0.789 (0.011)
FordB	0.606	0.667	0.653	0.917	0.711	0.662	0.804	0.688	0.599	0.807	0.711	0.728	0.697 (0.014)
GunPoint	0.913	0.980	0.987	1.000	0.994	0.993	1.000	0.973	0.987	1.000	0.993	0.947	0.947 (0.007)
Ham	0.600	0.476	0.552	0.667	0.667	0.571	0.648	0.743	0.762	0.686	0.562	0.648	0.689 (0.009)
HandOutlines	0.862	0.868	0.865	0.481	0.903	0.889	0.919	0.919	0.854	0.932	0.881	0.811	0.903 (0.005)
Haptics	0.370	0.399	0.399	0.468	0.461	0.393	0.523	0.445	0.490	0.523	0.432	0.393	0.783 (0.007)
Herring	0.516	0.547	0.547	0.625	0.547	0.578	0.625	0.609	0.641	0.672	0.578	0.531	0.609 (0.018)
InlineSkate	0.342	0.562	0.509	0.438	0.516	0.460	0.495	0.376	0.385	0.373	0.500	0.189	0.816 (0.004)
InsWngSnd	0.562	0.355	0.473	0.606	0.523	0.595	0.653	0.633	0.625	0.627	0.551	0.489	0.931 (0.001)
ItalyPrDmd	0.955	0.950	0.951	0.960	0.909	0.962	0.961	0.960	0.883	0.948	0.923	0.917	0.958 (0.008)
LrgKitApp	0.493	0.795	0.795	0.701	0.765	0.811	0.845	0.571	0.528	0.859	0.717	0.560	0.637 (0.015)
Lightning2	0.754	0.869	0.869	0.820	0.810	0.885	0.869	0.803	0.738	0.738	0.819	0.705	0.702 (0.026)
Lightning7	0.575	0.671	0.657	0.795	0.666	0.767	0.808	0.753	0.726	0.726	0.739	0.644	0.901 (0.009)
Mallat	0.914	0.949	0.927	0.950	0.949	0.939	0.954	0.919	0.960	0.964	0.908	0.976	0.976 (0.005)
Meat	0.933	0.933	0.933	0.733	0.900	0.933	0.917	0.933	0.933	0.850	0.883	0.833	0.917 (0.054)
MedicalImages	0.684	0.737	0.745	0.664	0.718	0.742	0.758	0.755	0.705	0.670	0.746	0.624	0.945 (0.002)
MidPhxAgeGp	0.519	0.539	0.500	0.571	0.545	0.558	0.636	0.578	0.578	0.643	0.487	0.545	0.660 (0.021)
MidPhxCorr	0.766	0.732	0.742	0.780	0.780	0.784	0.804	0.828	0.814	0.794	0.773	0.729	0.779 (0.009)
MidPhxTW	0.513	0.487	0.500	0.506	0.545	0.513	0.571	0.565	0.597	0.519	0.526	0.532	0.855 (0.006)
MoteStrain	0.879	0.833	0.768	0.883	0.846	0.883	0.937	0.869	0.903	0.897	0.922	0.777	0.848 (0.022)
NonInv_Thor1	0.829	0.806	0.841	0.259	0.838	0.846	0.931	0.876	0.842	0.950	0.812	0.710	0.997 (0.000)
NonInv_Thor2	0.880	0.893	0.890	0.770	0.900	0.913	0.946	0.910	0.862	0.951	0.841	0.754	0.997 (0.000)
OliveOil	0.867	0.833	0.867	0.167	0.867	0.867	0.900	0.867	0.833	0.900	0.867	0.733	0.750 (0.000)
OSULeaf	0.521	0.880	0.884	0.777	0.955	0.806	0.967	0.583	0.760	0.967	0.740	0.678	0.853 (0.005)
PhaOutCorr	0.761	0.739	0.761	0.765	0.772	0.773	0.770	0.803	0.830	0.763	0.756	0.744	0.784 (0.019)
Phoneme	0.109	0.269	0.268	0.218	0.265	0.305	0.349	0.212	0.276	0.321	0.237	0.174	0.963 (0.000)
ProxPhxAgeGp	0.785	0.800	0.795	0.834	0.834	0.805	0.854	0.849	0.849	0.844	0.795	0.780	0.893 (0.006)

Table 2. *Cont.*

Dataset	ED	DDTW	DTD	LCSS	BOSS	EE	FCOTE	TSF	TSBF	ST	LPS	FS	LA-ESN
ProxPhxCorr	0.808	0.794	0.794	0.849	0.849	0.808	0.869	0.828	0.873	0.883	0.842	0.804	0.865 (0.026)
ProxPhxTW	0.707	0.769	0.771	0.776	0.800	0.766	0.780	0.815	0.810	0.805	0.732	0.702	0.926 (0.007)
RefDev	0.395	0.445	0.445	0.515	0.499	0.437	0.547	0.589	0.472	0.581	0.459	0.333	0.596 (0.009)
ScreenType	0.360	0.429	0.437	0.429	0.464	0.445	0.547	0.456	0.509	0.520	0.416	0.413	0.593 (0.013)
ShapesAll	0.752	0.850	0.838	0.768	0.908	0.867	0.892	0.792	0.185	0.842	0.873	0.580	0.993 (0.000)
SmlKitApp	0.344	0.640	0.648	0.664	0.725	0.696	0.776	0.811	0.672	0.792	0.712	0.333	0.619 (0.023)
SonyAIBORobot	0.696	0.742	0.710	0.810	0.895	0.704	0.845	0.787	0.795	0.844	0.774	0.686	0.730 (0.054)
SonyAIBORobot2	0.859	0.892	0.892	0.875	0.888	0.878	0.952	0.810	0.778	0.934	0.872	0.790	0.836 (0.014)
StarlightCurves	0.849	0.962	0.962	0.947	0.978	0.926	0.980	0.969	0.977	0.979	0.963	0.918	0.960 (0.002)
Strawberry	0.946	0.954	0.957	0.911	0.976	0.946	0.951	0.965	0.954	0.962	0.962	0.903	0.947 (0.016)
SwedishLeaf	0.789	0.901	0.896	0.907	0.922	0.915	0.955	0.914	0.915	0.928	0.920	0.768	0.984 (0.001)
Symbols	0.899	0.953	0.963	0.932	0.961	0.959	0.963	0.915	0.946	0.882	0.963	0.934	0.950 (0.009)
SynthCntr	0.880	0.993	0.997	0.997	0.967	0.990	1.000	0.987	0.994	0.983	0.980	0.910	0.990 (0.003)
Trace	0.760	1.000	0.990	1.000	1.000	0.990	1.000	0.990	0.980	1.000	0.980	1.000	0.908 (0.010)
TwoLeadECG	0.747	0.978	0.985	0.996	0.985	0.971	0.993	0.759	0.866	0.997	0.948	0.924	0.795 (0.050)
TwoPatterns	0.907	1.000	0.999	0.993	0.991	1.000	1.000	0.991	0.976	0.955	0.982	0.908	0.980 (0.006)
UWaveX	0.739	0.779	0.775	0.791	0.753	0.805	0.822	0.804	0.831	0.803	0.829	0.695	0.941 (0.001)
UWaveY	0.662	0.716	0.698	0.703	0.661	0.726	0.759	0.727	0.736	0.730	0.761	0.596	0.922 (0.001)
UWaveZ	0.649	0.696	0.679	0.747	0.695	0.724	0.750	0.743	0.772	0.748	0.768	0.638	0.925 (0.001)
Wafer	0.995	0.980	0.993	0.996	0.995	0.997	1.000	0.996	0.995	1.000	0.997	0.997	0.995 (0.000)
Wine	0.611	0.574	0.611	0.500	0.912	0.574	0.648	0.630	0.611	0.796	0.629	0.759	0.497 (0.007)
WordSynonyms	0.618	0.730	0.730	0.607	0.659	0.779	0.757	0.647	0.688	0.570	0.755	0.431	0.970 (0.000)
Yoga	0.830	0.856	0.856	0.834	0.918	0.879	0.877	0.859	0.819	0.818	0.869	0.695	0.844 (0.004)
Average	0.703	0.766	0.767	0.756	0.805	0.785	0.831	0.780	0.769	0.814	0.777	0.694	0.861
Total	1	3	2	5	9	4	13	4	7	13	3	1	36
MR	10.605	7.855	7.947	7.026	5.474	6.158	2.961	6.224	6.566	4.750	7.263	10.921	4.684
ME	0.079	0.065	0.065	0.062	0.050	0.060	0.046	0.058	0.059	0.048	0.060	0.077	0.052

To better compare multiple classifiers on multiple datasets, we applied the pairwise post hoc analysis proposed by Benavoli et al. [39], where mean rank comparisons were computed using the Wilcoxon signature rank test [40] corrected for Holm's alpha (5%) [41]. A critical difference diagram [42] is used to depict the outcome. The more right the classifier is, the better it is. As illustrated in Figure 6, the classifiers connected by thick lines reflect no significant difference in accuracy between the two classifiers. Figure 6 shows the pairwise statistical difference comparison between LA-ESN and traditional classifiers and traditional classifiers. As can be seen from Figure 6, our model is located to the right of the majority of approaches.

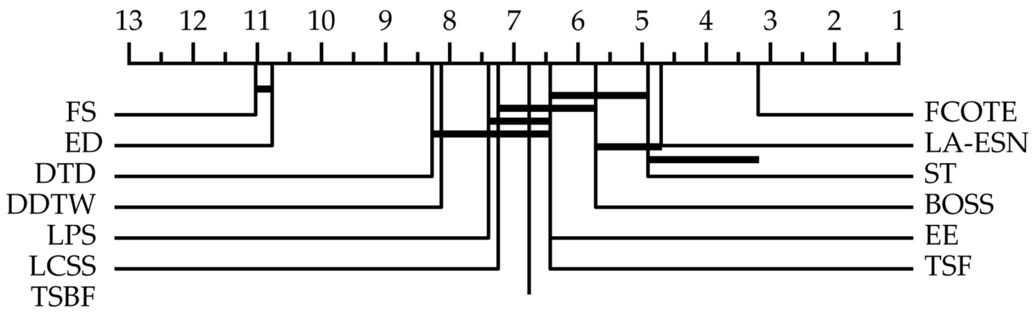

Figure 6. Pairwise statistical difference comparison of LA-ESN and 12 traditional classifiers.

4.3.2. Compared with Deep Learning Methods

In this section, we compared the LA-ESN model with four deep learning classifiers: MLP [2], FCN [2], ResNet [2] and Inception Time [32]. The following is a brief description of the four deep-learning comparison methods mentioned above:

(1) MLP (Multilayer Perceptrons). The final result is obtained by using a softmax layer and three fully connected layers of 500 cells. Dropout and ReLU are used to activate the model.

(2) FCN (Fully Convolutional Networks). The FCN model stacks three one-dimensional convolutional blocks with 128, 256 and 128 and kernel sizes of 3, 5 and 8. Then the features are fed into the global average pooling layer and the softmax layer

to obtain the final result. The FCN model uses the ReLU activation function and batch normalization.

(3) ResNet (Residual Network). The residual network is stacked with three residual blocks. Each residual block consists of 64, 128 and 256 convolutions of sizes 8, 5 and 3, respectively, followed by a ReLU activation function and batch normalization. ResNet extends the neural network to a profound structure by adding shortcut connections in each residual block. ResNet has a higher proclivity for overfitting the training data.

(4) Inception Time (AlexNet). Instead of the usual fully connected layer, it consists of two distinct residual blocks, each made of three Inception sub-blocks. The input of each residual block is transferred to the information of the next block via a fast linear connection, thus alleviating the problem of gradient disappearance by allowing a direct flow of gradients.

The experimental results of the proposed LA-ESN and these four classifiers on 128 datasets are shown in Table 3.

Table 3. The results of LA-ESN and four deep learning methods.

Dataset	MLP	FCN	ResNet	Inception Time	LA-ESN	
					AC(SD)	**Time(s)**
ACSF1	0.558	0.898	0.916	0.896	0.907 (0.004)	169.4
Adiac	0.391	0.841	0.833	0.830	0.984 (0.002)	348.8
AllGestureX	0.477	0.713	0.741	0.772	0.904 (0.001)	257.7
AllGestureY	0.571	0.784	0.794	0.813	0.912 (0.002)	257.0
AllGestureZ	0.439	0.692	0.726	0.792	0.891 (0.001)	257.7
ArrowHead	0.784	0.843	0.838	0.847	0.850 (0.016)	39.0
Beef	0.713	0.680	0.753	0.687	0.935 (0.009)	40.1
BeetleFly	0.880	0.910	0.850	0.800	0.790 (0.086)	85.1
BirdChicken	0.740	0.940	0.880	0.950	0.790 (0.037)	30.8
BME	0.905	0.836	0.999	0.993	0.928 (0.004)	29.6
Car	0.783	0.913	0.917	0.890	0.931 (0.002)	107.4
CBF	0.869	0.994	0.996	0.998	0.900 (0.033)	129.0
Chinatown	0.872	0.980	0.978	0.983	0.706 (0.091)	39.3
ChlorineCon	0.800	0.817	0.853	0.873	0.911 (0.009)	892.9
CinCECGTorso	0.838	0.829	0.838	0.842	0.909 (0.004)	1033.2
Coffee	0.993	1.000	1.000	1.000	1.000 (0.000)	94.7
Computers	0.558	0.819	0.806	0.786	0.548 (0.010)	299.5
CricketX	0.591	0.794	0.799	0.841	0.930 (0.004)	308.2
CricketY	0.598	0.793	0.810	0.839	0.933 (0.002)	323.2
CricketZ	0.629	0.810	0.809	0.849	0.935 (0.003)	302.1
Crop	0.618	0.738	0.743	0.751	0.911 (0.136)	1657.0
DiatomSizeR	0.909	0.346	0.301	0.935	0.978 (0.005)	108.2
DistPhxAgeGp	0.647	0.718	0.718	0.734	0.803 (0.007)	123.5
DistlPhxOutCorr	0.727	0.760	0.770	0.768	0.748 (0.009)	197.1
DistPhxTW	0.610	0.695	0.663	0.665	0.893 (0.005)	123.7
DodgerLoopDay	0.160	0.143	0.150	0.150	0.875 (0.013)	49.8
DodgerLoopGame	0.865	0.768	0.710	0.854	0.810 (0.048)	34.5
DodgerLoopWnd	0.978	0.904	0.952	0.970	0.971 (0.022)	34.1
Earthquakes	0.727	0.725	0.712	0.742	0.738 (0.012)	287.4
ECG200	0.914	0.888	0.874	0.918	0.904 (0.010)	93.7
ECG5000	0.930	0.940	0.935	0.939	0.973 (0.000)	712.0
ECGFiveDays	0.973	0.985	0.966	1.000	0.892 (0.038)	203.2
ElectricDevices	0.593	0.706	0.728	0.709	0.896 (0.003)	4303.5
EOGHorSignal	0.432	0.565	0.599	0.588	0.899 (0.005)	511.0
EOGVerticalSignal	0.418	0.446	0.445	0.464	0.899 (0.003)	508.3
EthanolLevel	0.386	0.484	0.758	0.804	0.765 (0.030)	933.2

Table 3. *Cont.*

Dataset	MLP	FCN	ResNet	Inception Time	LA-ESN	
					AC(SD)	Time(s)
FaceAll	0.794	0.938	0.867	0.801	0.974 (0.002)	624.2
FaceFour	0.836	0.930	0.955	0.957	0.910 (0.011)	47.9
FacesUCR	0.831	0.943	0.954	0.964	0.977 (0.003)	320.9
FiftyWords	0.708	0.646	0.740	0.807	0.988 (0.001)	311.3
Fish	0.848	0.961	0.981	0.976	0.965 (0.002)	165.3
FordA	0.816	0.914	0.937	0.957	0.786 (0.010)	3204.3
FordB	0.707	0.772	0.813	0.849	0.700 (0.014)	2970.7
FreezerRegularT	0.906	0.997	0.998	0.996	0.959 (0.013)	352.7
FreezerSmallTrain	0.686	0.683	0.832	0.866	0.676 (0.007)	324.5
Fungi	0.863	0.018	0.177	1.000	0.986 (0.006)	35.4
GestureMidAirD1	0.575	0.695	0.698	0.732	0.969 (0.002)	111.8
GestureMidAirD2	0.545	0.631	0.668	0.708	0.963 (0.002)	110.5
GestureMidAirD3	0.382	0.326	0.340	0.366	0.955 (0.002)	111.3
GesturePebbleZ1	0.792	0.880	0.901	0.922	0.942 (0.003)	100.8
GesturePebbleZ2	0.701	0.781	0.777	0.875	0.922 (0.018)	104.3
GunPoint	0.928	1.000	0.991	1.000	0.947 (0.007)	52.7
GunPointAgeSpan	0.934	0.996	0.997	0.987	0.882 (0.009)	71.4
GunPointMaleFe	0.980	0.997	0.992	0.996	0.985 (0.004)	70.1
GunPointOldYg	0.941	0.989	0.989	0.962	0.992 (0.007)	70.6
Ham	0.699	0.707	0.758	0.705	0.688 (0.009)	119.4
HandOutlines	0.914	0.799	0.914	0.946	0.903 (0.005)	3273.4
Haptics	0.425	0.490	0.510	0.549	0.783 (0.007)	368.5
Herring	0.491	0.644	0.600	0.666	0.613 (0.018)	89.4
HouseTwenty	0.734	0.982	0.983	0.975	0.803 (0.009)	154.8
InlineSkate	0.335	0.332	0.377	0.485	0.816 (0.004)	631.9
InsectEPGRegTra	0.646	0.999	0.998	0.998	0.788 (0.015)	94.3
InsectEPGSmallTra	0.627	0.218	0.372	0.941	0.756 (0.005)	73.0
InsectWingSnd	0.604	0.392	0.499	0.630	0.931 (0.001)	497.3
ItalyPowerDemand	0.953	0.963	0.962	0.964	0.958 (0.008)	166.9
LargeKitchenApp	0.470	0.903	0.901	0.900	0.641 (0.014)	497.1
Lightning2	0.682	0.734	0.780	0.787	0.702 (0.028)	112.5
Lightning7	0.616	0.825	0.827	0.803	0.901 (0.009)	86.6
Mallat	0.923	0.967	0.974	0.941	0.976 (0.005)	1212.1
Meat	0.893	0.803	0.990	0.933	0.916 (0.059)	93.0
MedicalImages	0.719	0.778	0.770	0.787	0.945 (0.002)	253.2
MelbournePed	0.863	0.912	0.909	0.908	0.973 (0.004)	322.6
MidPhxOutAgeGp	0.522	0.535	0.545	0.523	0.660 (0.021)	143.6
MidPhxOutCorr	0.755	0.795	0.826	0.816	0.781 (0.008)	208.5
MidPhxTW	0.536	0.501	0.495	0.508	0.855 (0.006)	173.7
MixedRegularTrain	0.907	0.955	0.973	0.966	0.963 (0.002)	1101.6
MixedSmallTrain	0.841	0.893	0.917	0.912	0.934 (0.004)	774.1
MoteStrain	0.855	0.936	0.924	0.886	0.848 (0.022)	201.1
NonFetalECGTh1	0.915	0.958	0.941	0.956	0.997 (0.000)	2374.7
NonFetalECGTh2	0.918	0.953	0.944	0.958	0.997 (0.000)	2320.2
OliveOil	0.653	0.720	0.847	0.820	0.750 (0.000)	63.8
OSULeaf	0.560	0.979	0.980	0.925	0.853 (0.005)	201.5
PhaOutCorr	0.756	0.818	0.845	0.838	0.777 (0.013)	512.4
Phoneme	0.094	0.328	0.333	0.328	0.963 (0.000)	1109.9
PickupGestureWZ	0.604	0.744	0.704	0.744	0.936 (0.006)	31.8
PigAirwayPressure	0.065	0.172	0.406	0.532	0.969 (0.001)	302.6
PigArtPressure	0.105	0.987	0.991	0.993	0.971 (0.001)	303.6
PigCVP	0.076	0.831	0.918	0.953	0.974 (0.000)	302.1
PLAID	0.625	0.904	0.940	0.937	0.932 (0.001)	791.3
Plane	0.977	1.000	1.000	1.000	0.993 (0.001)	79.7
PowerCons	0.977	0.863	0.879	0.948	0.979 (0.010)	70.1
ProxPhxOutAgeGp	0.849	0.825	0.847	0.845	0.893 (0.006)	127.1
ProxPhaxOutCorr	0.730	0.907	0.920	0.918	0.865 (0.026)	183.3

Table 3. *Cont.*

Dataset	MLP	FCN	ResNet	Inception Time	LA-ESN	
					AC(SD)	Time(s)
ProxPhxTW	0.767	0.761	0.773	0.781	0.926 (0.007)	133.6
RefDevices	0.377	0.497	0.530	0.523	0.596 (0.009)	439.7
Rock	0.852	0.632	0.552	0.752	0.917 (0.029)	105.3
ScreenType	0.402	0.622	0.615	0.580	0.591 (0.014)	470.7
SemgHandGenCh2	0.822	0.816	0.824	0.802	0.825 (0.033)	599.5
SemgHandMovCh2	0.435	0.476	0.439	0.420	0.815 (0.015)	710.8
SemgHandSubCh2	0.817	0.742	0.739	0.787	0.922 (0.007)	707.8
ShakeGestureWZ	0.548	0.884	0.880	0.900	0.920 (0.008)	33.2
ShapeletSim	0.513	0.706	0.782	0.917	0.510 (0.032)	54.4
ShapesAll	0.776	0.894	0.926	0.918	0.993 (0.000)	561.3
SmallKitchenApp	0.380	0.777	0.781	0.756	0.615 (0.024)	452.6
SmoothSubspace	0.980	0.975	0.980	0.981	0.880 (0.021)	30.4
SonyAIBORobSur1	0.692	0.958	0.961	0.864	0.734 (0.059)	116.2
SonyAIBORobSur2	0.831	0.980	0.975	0.946	0.838 (0.014)	153.8
StarLightCurves	0.950	0.965	0.972	0.978	0.960 (0.002)	4779.5
Strawberry	0.959	0.975	0.980	0.983	0.947 (0.016)	350.3
SwedishLeaf	0.845	0.967	0.963	0.964	0.984(0.001)	247.5
Symbols	0.836	0.955	0.893	0.980	0.948 (0.009)	345.0
SyntheticControl	0.973	0.989	0.997	0.996	0.990 (0.003)	100.6
ToeSegmentation1	0.589	0.961	0.957	0.961	0.639 (0.025)	52.9
ToeSegmentation2	0.745	0.889	0.894	0.943	0.794 (0.026)	44.2
Trace	0.806	1.000	1.000	1.000	0.910 (0.009)	78.2
TwoLeadECG	0.753	0.999	1.000	0.997	0.806 (0.048)	190.0
TwoPatterns	0.948	0.870	1.000	1.000	0.980 (0.006)	813.6
UMD	0.949	0.988	0.990	0.982	0.925 (0.021)	33.0
UWaveAll	0.954	0.818	0.861	0.944	0.988 (0.001)	1740.2
UWaveX	0.768	0.754	0.781	0.814	0.941 (0.001)	1324.3
UWaveY	0.699	0.642	0.666	0.755	0.922 (0.001)	1346.0
UWaveZ	0.697	0.727	0.749	0.750	0.925 (0.001)	1251.3
Wafer	0.996	0.997	0.998	0.999	0.995 (0.000)	1309.3
Wine	0.541	0.611	0.722	0.659	0.496 (0.007)	64.2
WordSynonyms	0.599	0.561	0.617	0.732	0.970 (0.000)	296.5
Worms	0.457	0.782	0.761	0.769	0.794 (0.012)	176.6
WormsTwoClass	0.608	0.743	0.748	0.782	0.626 (0.056)	177.2
Yoga	0.856	0.837	0.867	0.891	0.845 (0.004)	900.2
Average	0.705	0.786	0.807	0.836	0.871	
Total	2	13	25	32	64	
MR	4.313	3.234	2.648	2.250	2.430	
ME	0.066	0.048	0.043	0.037	0.047	

From Table 3, we can obtain the following: (1) On 64 datasets, LA-ESN has the highest classification accuracy. Furthermore, the total number significantly outperforms other methods. (2) The average classification accuracy of all datasets of the proposed LA-ESN is also superior to other compared methods. (3) The *MR* value of the proposed LA-ESN is 2.430, which is higher than the Inception Time method but smaller than the other three classifiers. (4) The proposed LA-ESN has a *ME* value of 0.047, which is lower than the MLP and FCN but greater than ResNet and Inception Time. (5) The performances of LA-ESN and Inception Time perform better than the other three methods. This reason is that LA-ESN and Inception Time can efficiently handle temporal dependencies in time series with high nonlinear mapping capacity and dynamic memory. (6) When the epoch is set as 500, the running time of the proposed LA-ESN on small datasets is concise while is acceptable on large datasets. A paired statistical difference comparison of the five deep learning methods is shown in Figure 7. As seen in the diagram, our model is to the right of the majority of methods.

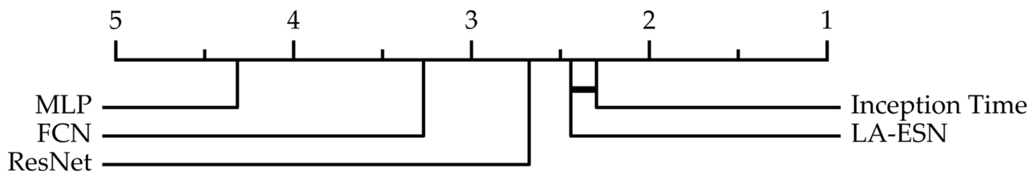

Figure 7. Pairwise statistical difference comparison of LA-ESN and four deep learning methods.

4.3.3. Ablation Study

In this subsection, a step-by-step exploration has been constructed further to verify the validity of the proposed LA-ESN model. We divided the model into four modules: ESN-LSTM, ESN-CNN, ESN_LSTM_ATT (ELA) and ESN_CNN_ATT (ECA), and conducted experiments on 17 datasets. These four methods first employ the ESN to perform representation learning on time series and then use different classifiers such as LSTM, CNN, LSTM with attention (LSTM_ATT) and CNN with attention (CNN_ATT) to extract feature information from the representation. In order to verify the validity of each module, a unified setting is adopted for the parameters. In ESN, the spectral radius *SR* is 0.9, the input cell scale *IS* is 0.1, the reservoir sparsity *SP* is 0.7, and the size of the reservoir is 32. The epoch of the whole experiment is 500, and the batch size is 25. LSTM_ATT can automatically record long-term temporal dependencies of sequences, and CNN_ATT can extract feature information from different scales. LA-ESN combines both advantages to extract more valuable information from datasets and achieve better classification accuracy. Table 4 displays the outcomes of LA-ESN and four distinct classifiers on 17 datasets. On 13 datasets, it can be seen that LA-ESN produces the best results. At the same time, we build the critical difference diagram shown in Figure 8.

Table 4. Results of LA-ESN and four ablation models.

Dataset	**ESN_CNN**	**ECA**	**ESN_LSTM**	**ELA**	**LA-ESN**
Adiac	0.952	0.978	0.983	0.974	0.984
Beef	0.826	0.906	0.899	0.846	0.947
CricketX	0.930	0.925	0.928	0.925	0.930
CricketY	0.932	0.927	0.929	0.917	0.933
ECG5000	0.973	0.973	0.967	0.970	0.973
HandOutlines	0.854	0.856	0.878	0.899	0.908
Haptics	0.736	0.784	0.753	0.779	0.783
NonInv_Thor2	0.995	0.990	0.995	0.992	0.997
ProxPhxAgeGp	0.852	0.869	0.874	0.871	0.893
ProxPhxTW	0.913	0.926	0.934	0.935	0.926
ShapesAll	0.992	0.992	0.990	0.987	0.993
SwedishLeaf	0.982	0.983	0.978	0.978	0.984
UWaveX	0.942	0.941	0.933	0.934	0.941
UWaveY	0.910	0.915	0.899	0.912	0.922
UWaveZ	0.924	0.927	0.907	0.915	0.925
Wafer	0.986	0.960	0.990	0.990	0.995
WordSynonyms	0.967	0.967	0.966	0.964	0.970
Average	0.922	0.931	0.930	0.929	0.941
Total	3	3	0	1	13
MR	3.235	2.765	3.412	3.588	1.294
ME	0.016	0.015	0.014	0.014	0.012

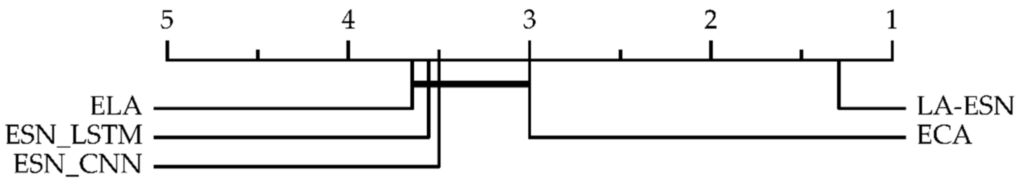

Figure 8. Pairwise statistical difference comparison of LA-ESN and four ablation models.

5. Conclusions

This study proposes a deep learning model called LA-ESN for the end-to-end classification of univariate time series. The original time series is first passed through the ESN model in LA-ESN to obtain the echo state representation matrix. Then, the echo state representation matrix is used as the input of the LSTM module and the multi-scale convolutional module, and feature extraction operations are performed on them. Finally, the results from the two modules are concatenated, and the softmax function is utilized to produce the final classification results. The attention-based LSTM module can automatically record the long-term time dependence of the sequences, and the multi-scale 1D convolutional attention module can extract feature information from echo state representations at different scales, highlighting the spatial sparsity and heterogeneity of the data. Without any data reshaping or pre-processing, the model performs better. Based on extensive trials on the UCR time series dataset, the proposed LA-ESN model outperforms several older approaches and current popular deep learning methods in the great majority of selected datasets. Experiments suggest that our technique may perform better on certain datasets.

However, this experiment still needs to be improved in studying the class imbalance problem. As a consequence, in the future, we will improve the model to obtain the best results on most datasets while also addressing the class imbalance issue. Second, we can consider modifying the model to accomplish the classification of multivariate time series data. Finally, LA-ESN is a more generic model in the field of time series, and we can subsequently consider using LA-ESN for tasks such as time series prediction and clustering.

Author Contributions: Data curation, H.S., M.L., J.H., P.L. and Y.P.; Formal analysis, H.S., M.L. and Y.P.; Methodology, H.S., M.L. and Y.Y.; Resources, H.S., M.L. and J.H.; Supervision, P.L. and Y.Y.; Writing—original draft, H.S., M.L. and P.L.; Writing—review and editing, H.S., M.L., P.L. and Y.Y. All authors have read and agreed to the published version of the manuscript.

Funding: This work is supported in part by grants from the National Natural Science Foundation of China (Nos. 62062040, 61967010 and 62067003), the Outstanding Youth Project of Jiangxi Natural Science Foundation (No. 20212ACB212003), the Jiangxi Province Key Subject Academic and Technical Leader Funding Project (No. 20212BCJ23017), the Jiangxi Province Graduate Innovation Project (No. YC2022-S275), the Central Guided Local Science and Technology Development Special Project (20222ZDH04090) and the Jiangxi Provincial Social Science "14th Five-Year" Planning Project (22WT76).

Data Availability Statement: The data were derived from public domain resources.

Conflicts of Interest: The authors declare no conflict of interest.

References

1. Hamilton, J.D. *Time Series Analysis*; Princeton University Press: Oxford, UK, 2020.
2. Ismail Fawaz, H.; Forestier, G.; Weber, J.; Idoumghar, L.; Muller, P.-A. Deep learning for time series classification: A review. *Data Min. Knowl. Discov.* **2019**, *33*, 917–963. [CrossRef]
3. Xiao, X.; Li, R.; Zheng, H.-T.; Ye, R.; Kumar Sangaiah, A.; Xia, S. Novel dynamic multiple classification system for network traffic. *Inf. Sci.* **2018**, *479*, 526–541. [CrossRef]
4. Michida, R.; Katayama, D.; Seiji, I.; Wu, Y.; Koide, T.; Tanaka, S.; Okamoto, Y.; Mieno, H.; Tamaki, T.; Yoshida, S. A Lesion Classification Method Using Deep Learning Based on JNET Classification for Computer-Aided Diagnosis System in Colorectal Magnified NBI Endoscopy. In Proceedings of the 36th International Technical Conference on Circuits/Systems, Computers and Communications (ITC-CSCC), Jeju, Republic of Korea, 27–30 June 2021; pp. 1–4.

5. Mori, U.; Mendiburu, A.; Miranda, I.; Lozano, J. Early classification of time series using multi-objective optimization techniques. *Inf. Sci.* **2019**, *492*, 204–218. [CrossRef]
6. Wan, Y.; Si, Y.-W. A formal approach to chart patterns classification in financial time series. *Inf. Sci.* **2017**, *411*, 151–175. [CrossRef]
7. Wang, H.; Wu, Q.J.; Wang, D.; Xin, J.; Yang, Y.; Yu, K. Echo state network with a global reversible autoencoder for time series classification. *Inf. Sci.* **2021**, *570*, 744–768. [CrossRef]
8. Marteau, P.-F.; Gibet, S. On Recursive Edit Distance Kernels with Application to Time Series Classification. *IEEE Trans. Neural Networks Learn. Syst.* **2014**, *26*, 1121–1133. [CrossRef]
9. Wang, J.; Zhao, Y. Time Series K-Nearest Neighbors Classifier Based on Fast Dynamic Time Warping. In Proceedings of the 2021 IEEE International Conference on Artificial Intelligence and Computer Applications (ICAICA), Dalian, China, 28–30 June 2021; pp. 751–754.
10. Gustavo, E.; Batista, A.; Keogh, E.J.; Tataw, O.M.; De Souza, V.M.A. CID: An efficient complexity-invariant distance for time series. *Data Min. Knowl. Discov.* **2013**, *28*, 634–669. [CrossRef]
11. Purnawirawan, A.; Wibawa, A.D.; Wulandari, D.P. Classification of P-wave Morphology Using New Local Distance Transform and Random Forests. In Proceedings of the 6th International Conference on Science and Technology (ICST), Yogyakarta, Indonesia, 7–8 September 2020; pp. 1–6.
12. Qiao, Z.; Mizukoshi, Y.; Moteki, T.; Iwata, H. Operation State Identification Method for Unmanned Construction: Extended Search and Registration System of Novel Operation State Based on LSTM and DDTW. In Proceedings of the IEEE/SICE International Symposium on System Integration (SII), Virtual Conference, 9–12 January 2022; pp. 13–18.
13. Rahman, B.; Warnars, H.L.H.S.; Sabarguna, B.S.; Budiharto, W. Heart Disease Classification Model Using K-Nearest Neighbor Algorithm. In Proceedings of the 6th International Conference on Informatics and Computing (ICIC), Virtual Conference, 3–4 November 2021; pp. 1–4.
14. Górecki, T. Using derivatives in a longest common subsequence dissimilarity measure for time series classification. *Pattern Recognit. Lett.* **2014**, *45*, 99–105. [CrossRef]
15. Baydogan, M.G.; Runger, G.; Tuv, E. A Bag-of-Features Framework to Classify Time Series. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 2796–2802. [CrossRef]
16. Deng, H.; Runger, G.; Tuv, E.; Vladimir, M. A time series forest for classification and feature extraction. *Inf. Sci.* **2013**, *239*, 142–153. [CrossRef]
17. Ji, C.; Zhao, C.; Liu, S.; Yang, C.; Pan, L.; Wu, L.; Meng, X. A fast shapelet selection algorithm for time series classification. *Comput. Networks* **2019**, *148*, 231–240. [CrossRef]
18. Arul, M.; Kareem, A. Applications of shapelet transform to time series classification of earthquake, wind and wave data. *Eng. Struct.* **2021**, *228*, 111564. [CrossRef]
19. Baydogan, M.G.; Runger, G. Time series representation and similarity based on local autopatterns. *Data Min. Knowl. Discov.* **2016**, *30*, 476–509. [CrossRef]
20. Kate, R.J. Using dynamic time warping distances as features for improved time series classification. *Data Min. Knowl. Discov.* **2016**, *30*, 283–312. [CrossRef]
21. Karim, F.; Majumdar, S.; Darabi, H.; Harford, S. Multivariate LSTM-FCNs for time series classification. *Neural Networks* **2019**, *116*, 237–245. [CrossRef]
22. Ma, Q.; Chen, E.; Lin, Z.; Yan, J.; Yu, Z.; Ng, W.W.Y. Convolutional Multitimescale Echo State Network. *IEEE Trans. Cybern.* **2019**, *51*, 1613–1625. [CrossRef]
23. Karim, F.; Majumdar, S.; Darabi, H. Insights into LSTM Fully Convolutional Networks for Time Series Classification. *IEEE Access* **2019**, *7*, 67718–67725. [CrossRef]
24. Koh, B.H.D.; Lim, C.L.P.; Rahimi, H.; Woo, W.L.; Gao, B. Deep Temporal Convolution Network for Time Series Classification. *Sensors* **2021**, *21*, 603. [CrossRef]
25. Ku, B.; Kim, G.; Ahn, J.-K.; Lee, J.; Ko, H. Attention-Based Convolutional Neural Network for Earthquake Event Classification. *IEEE Geosci. Remote. Sens. Lett.* **2020**, *18*, 2057–2061. [CrossRef]
26. Tripathi, A.M.; Baruah, R.D. Multivariate Time Series Classification with An Attention-Based Multivariate Convolutional Neural Network. In Proceedings of the 2020 International Joint Conference on Neural Networks (IJCNN), Glasgow, UK, 19–24 July 2020; pp. 1–8.
27. Sun, J.; Takeuchi, S.; Yamasaki, I. Prototypical Inception Network with Cross Branch Attention for Time Series Classification. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–7.
28. Li, D.; Lian, C.; Yao, W. Research on time series classification based on convolutional neural network with attention mechanism. In Proceedings of the 11th International Conference on Intelligent Control and Information Processing (ICICIP), Yunnan, China, 3–7 December 2021; pp. 88–93.
29. Huang, Z.; Yang, C.; Chen, X.; Zhou, X.; Chen, G.; Huang, T.; Gui, W. Functional deep echo state network improved by a bi-level optimization approach for multivariate time series classification. *Appl. Soft Comput.* **2021**, *106*. [CrossRef]
30. Wang, H.; Liu, Y.; Wang, D.; Luo, Y.; Tong, C.; Lv, Z. Discriminative and regularized echo state network for time series classification. *Pattern Recognit.* **2022**, *130*, 1–14. [CrossRef]
31. Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; Zhao, J.L. Exploiting multi-channels deep convolutional neural networks for multivariate time series classification. *Front. Comput. Sci.* **2016**, *10*, 96–112. [CrossRef]

32. Fawaz, H.I.; Lucas, B.; Forestier, G.; Pelletier, C.; Schmidt, D.F.; Weber, J.; Webb, G.I.; Idoumghar, L.; Muller, P.-A.; Petitjean, F. InceptionTime: Finding AlexNet for time series classification. *Data Min. Knowl. Discov.* **2020**, *34*, 1936–1962. [CrossRef]
33. Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; Zhao, J.L. Time series classification using multi-channels deep convolutional neural networks. In Proceedings of the International Conference on Web-Age Information Management, Macau, China, 16–18 June 2014; Springer: Cham, Switzerland, 2014; pp. 298–310.
34. Dau, H.A.; Bagnall, A.; Kamgar, K.; Yeh, C.-C.M.; Zhu, Y.; Gharghabi, S.; Ratanamahatana, C.A.; Keogh, E. The UCR time series archive. *IEEE/CAA J. Autom. Sin.* **2019**, *6*, 1293–1305. [CrossRef]
35. Wang, Z.; Yan, W.; Oates, T. Time series classification from scratch with deep neural networks: A strong baseline. In Proceedings of the 2017 International Joint Conference on Neural Networks (IJCNN), Anchorage, AK, USA, 14–19 May 2017; pp. 1578–1585.
36. Schäfer, P. The BOSS is concerned with time series classification in the presence of noise. *Data Min. Knowl. Discov.* **2015**, *29*, 1505–1530. [CrossRef]
37. Lines, J.; Bagnall, A. Time series classification with ensembles of elastic distance measures. *Data Min. Knowl. Discov.* **2015**, *29*, 565–592. [CrossRef]
38. Bagnall, A.; Lines, J.; Hills, J.; Bostrom, A. Time-Series Classification with COTE: The Collective of Transformation-Based Ensembles. *IEEE Trans. Knowl. Data Eng.* **2015**, *27*, 2522–2535. [CrossRef]
39. Benavoli, A.; Corani, G.; Mangili, F. Should we really use post-hoc tests based on mean-ranks? *J. Mach. Learn. Res.* **2016**, *17*, 152–161. [CrossRef]
40. Wilcoxon, F. Individual comparisons by ranking methods. In *Breakthroughs in Statistics*; Springer: New York, NY, USA, 1992; pp. 196–202.
41. Holm, S. A simple sequentially rejective multiple test procedure. *Scand. J. Stat.* **1979**, *6*, 65–70.
42. Demšar, J. Statistical comparisons of classifiers over multiple data sets. *J. Mach. Learn. Res.* **2006**, *7*, 1–30. [CrossRef]