Neural Networks 117 (2019) 225–239

Contents lists available at ScienceDirect

Neural Networks

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

Time series classification with Echo Memory Networks

Qianli Ma [a,*], Wanqing Zhuang [a], Lifeng Shen [a], Garrison W. Cottrell [b]

[a] School of Computer Science and Engineering, South China University of Technology, Guangzhou, China
[b] Department of Computer Science and Engineering, University of California, San Diego, USA

ARTICLE INFO

Article history:
Received 21 February 2019
Received in revised form 3 May 2019
Accepted 9 May 2019
Available online 28 May 2019

Keywords:
Echo state networks
Multi-scale convolution
Time series classification

ABSTRACT

Echo state networks (ESNs) are randomly connected recurrent neural networks (RNNs) that can be used as a temporal kernel for modeling time series data, and have been successfully applied on time series prediction tasks. Recently, ESNs have been applied to time series classification (TSC) tasks. However, previous ESN-based classifiers involve either training the model by predicting the next item of a sequence, or predicting the class label at each time step. The former is essentially a predictive model adapted from time series prediction work, rather than a model designed specifically for the classification task. The latter approach only considers local patterns at each time step and then averages over the classifications. Hence, rather than selecting the most discriminating sections of the time series, this approach will incorporate non-discriminative information into the classification, reducing accuracy. In this paper, we propose a novel end-to-end framework called the Echo Memory Network (EMN) in which the time series dynamics and multi-scale discriminative features are efficiently learned from an unrolled echo memory using multi-scale convolution and max-over-time pooling. First, the time series data are projected into the high dimensional nonlinear space of the reservoir and the echo states are collected into the echo memory matrix, followed by a single multi-scale convolutional layer to extract multi-scale features from the echo memory matrix. Max-over-time pooling is used to maintain temporal invariance and select the most important local patterns. Finally, a fully-connected hidden layer feeds into a softmax layer for classification. This architecture is applied to both time series classification and human action recognition datasets. For the human action recognition datasets, we divide the action data into five different components of the human body, and propose two spatial information fusion strategies to integrate the spatial information over them. With one training-free recurrent layer and only one layer of convolution, the EMN is a very efficient end-to-end model, and ranks first in overall classification ability on 55 TSC benchmark datasets and four 3D skeleton-based human action recognition tasks.

1. Introduction

Echo state networks (ESNs) are a popular approach for training recurrent neural networks (RNNs) efficiently. An ESN typically consists of three components: an input layer, a large RNN hidden layer (called the reservoir) and a linear output layer. The input-to-hidden and hidden-to-hidden (recurrent) weights are randomly initialized and fixed during the learning stage. By learning the state dynamics generated from the untrained recurrent hidden layer, ESNs can avoid the laborious process of gradient-descent RNN training, yet achieve excellent performance in time series prediction (Chatzis & Demiris, 2011; Jaeger & Haas, 2004; Li, Han, & Wang, 2012; Lukoševičius & Jaeger, 2009; Qiao, Li, Han, & Li, 2017; Shi & Han, 2007), speech recognition (Skowronski & Harris, 2007; Triefenbach, Jalalvand, Demuynck, & Martens, 2013) and signal processing (Xia, Jelfs, Hulle, Principe, & Mandic, 2011).

The most distinctive feature of ESNs is that the weights of the reservoir are randomly initialized and fixed during training. The reservoir can be regarded as a high-dimensional (usually 100–1000D) nonlinear temporal kernel, and automatically provides abundant echo state representations (ESRs) of the input time series. Moreover, due to the sparse connectivity of neurons in the reservoir, many loosely coupled oscillators are generated, and the information persists in one part of the reservoir without being propagated to other parts too quickly, which contributes to the short-term memory property of ESNs (Jaeger & Haas, 2004). Therefore, ESNs have been successfully applied to time series prediction tasks due to their high dimensional non-linear mapping capability and short-term memory.

Although ESNs have achieved great success in time series prediction tasks, the use of them in time series classifications

* Corresponding author.
E-mail addresses: qianlima@scut.edu.cn (Q. Ma), scutzwq@gmail.com (W. Zhuang), scuterlifeng@foxmail.com (L. Shen), gary@ucsd.edu (G.W. Cottrell).

https://doi.org/10.1016/j.neunet.2019.05.008
0893-6080/

(TSC) tasks has not been fully explored. In recent years, some researchers have applied ESNs to TSC tasks. In Skowronski and Harris (2007), the authors proposed a predictive ESN classifier and applied it to speech classification of ten digits. The predictive classifier trains different predictors for diverse classes and the test sample is classified into the category corresponding to the predictor that predicts it the best (Lukoševičius & Jaeger, 2009). In Chen, Tang, Tino, Cohn, and Yao (2015), the authors proposed the model-metric co-learning (MMCL) methodology for TSC tasks, which trains the output weights for each time series through a one-step prediction task, and then uses the output weights as a new feature representation for classification. These attempts at using ESNs to handle TSC tasks are essentially predictive models rather than classifiers, since these models are trained by predicting the next item of a sequence.

There is another kind of ESN-based classifier that directly classifies the input time series at each time step. In Lukoševičius (2012), Lukoševičius and Jaeger (2009) and Verstraeten, Schrauwen, dHaene, and Stroobandt (2007), the authors train the models to generate a class label at each time step and then use the average prediction as the final class probabilities. However, although using current echo state is adequate to predict the next item of a time series, using the echo state at a single time step is inadequate to obtain the final class label since the echo states at all time steps need to be considered in TSC tasks. More importantly, different classes of time series can often be distinguished by some important discriminative local patterns (Ye & Keogh, 2009). Although the echo state at each time step contains local temporal information due to the short-term memory property of ESNs, conducting classification at each time step will interfere with the performance of classifiers since some unimportant patterns (i.e., some local patterns exist in each class) are forced to determine the category. For example, waving an arm and drawing a circle both include the local pattern of raising a hand, but these insignificant local patterns cannot be used to determine the category label.

In recent years, deep learning architectures have been applied to a wide variety of tasks and achieved great success. Among them, Sukhbaatar, Weston, Fergus, et al. proposed a novel model called the end-to-end memory network, which projects the feature representation of each word of a sentence into a new feature space, and used the weighted sum of all the new word representations as the output memory representations (Sukhbaatar et al., 2015). The weighted sum process can be seen as choosing some important information from the representation at all time steps. Therefore, it is natural to expect that we can use an ESN as a memory encoder to obtain temporal features of time series over time steps, and then use a decoder to extract the discriminative features. Here, we use Convolutional Neural Networks (CNNs) (Lecun, Bottou, Bengio, & Haffner, 1998) as the decoder to replace the original linear regression, since CNNs can learn discriminative features from ESRs using back-propagation, and they are also efficiently computed due to weight tying and the convolution operation (Goodfellow, Bengio, Courville, & Bengio, 2016). More importantly, multi-scale temporal information naturally exists in time series (Cui, Chen, & Chen, 2016), which plays a pivotal role in dealing with TSC tasks. The original approach of linear regression cannot learn multi-scale discriminative features from the ESRs of all time steps, while multi-scale convolutional kernels (Simonyan & Zisserman, 2014) can. We also use max-over-time pooling to select the most discriminative local patterns that have the greatest impact on classification.

In this paper, we combine the advantages of echo state networks and convolutional neural networks, to propose a novel neural network model called the Echo Memory Network (EMN). An EMN consists of two phases: an encoding and a decoding phase. In the encoding phase, the raw input data are projected into a high dimensional nonlinear space to obtain the Echo-State representations (ESRs). In the decoding phase, the echo states are decoded using a multi-scale convolution layer and a max-over-time pooling layer, and then the probability distribution for each label is computed by a fully-connected hidden layer followed by a softmax. In contrast to traditional ESNs, EMN collects the ESRs of all time steps and stores them in a matrix we call the echo memory matrix, and the convolution and pooling operations are conducted on the entire echo memory matrix. The echo memory matrix contains abundant contextual information extracted by the reservoir. Multi-scale convolution over time can learn temporal features from the memory matrix and select the most discriminative patterns using max-over-time pooling. Since the weights of the reservoir are fixed, and only one convolutional layer is used, EMN is a very efficient end-to-end model. Our contributions can be summarized as follows.

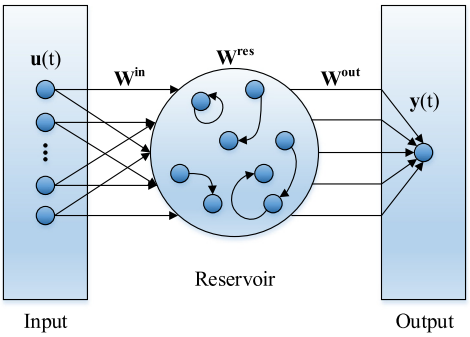


Fig. 1. General architecture of an Echo State Network.

1. We propose an efficient end-to-end model, which is able to extract discriminative temporal features from the reservoir representations.
2. By integrating the efficiency of ESNs in representing time series and the effectiveness of CNNs for multi-scale feature extraction into a unified framework, EMN provides an alternative to both the reservoir computing and the deep learning paradigms.
3. EMN is a universal TSC framework, which can be easily generalized to deal with other tasks such as 3D skeleton-based action recognition tasks. Our state-of-the-art performance on both the UCR time series classification benchmarks (Chen et al., 2015) and four 3D-skeleton-based action recognition tasks prove its effectiveness, and the visualization results demonstrate interpretability.

The remainder of the paper is organized as follows. Section 2 gives a brief introduction to ESNs. Section 3 reviews related work. Section 4 introduces the proposed EMN model in detail. In Section 5, experiments are conducted on TSC and human action recognition benchmarks and the results are analyzed. Section 6 concludes the paper.

2. Echo state networks

An ESN without output feedback consists of three basic components: an input layer, a large recurrent hidden layer (called the reservoir) and an output layer. The input layer is randomly connected to the reservoir. The reservoir contains sparse random connections. The only adaptable parameters are the output weights, which usually can be obtained by linear regression. The general architecture of an ESN is illustrated in Fig. 1.

Given a K-dimensional input $\mathbf{u}(t) \in \mathbb{R}^K$ at time step t and N-dimensional reservoir state $\mathbf{x}(t-1) \in \mathbb{R}^N$ at time step $t-1$, the update equations of entire system are as follows:

$$\mathbf{x}(t) = f(\mathbf{W}^{res}\mathbf{x}(t-1) + \mathbf{W}^{in}\mathbf{u}(t)) \tag{1}$$

$$\mathbf{y}(t) = f^{out}(\mathbf{W}^{out}\mathbf{x}(t)) \tag{2}$$

where $\mathbf{y}(t) \in \mathbb{R}^L$ is the L-dimensional output at time step t. $\mathbf{W}^{in}$, $\mathbf{W}^{res}$, $\mathbf{W}^{out}$ denote the connection weights from the input layer to the reservoir layer, the reservoir to itself and the reservoir to the output layer, respectively. $\mathbf{W}^{in}$, $\mathbf{W}^{res}$ are initialized randomly and fixed. Only $\mathbf{W}^{out}$ is adaptable. f is the activation function in the reservoir (usually $tanh(\cdot)$) and f^{out} is the activation function of the output layer (usually $identity(\cdot)$).

There are three hyperparameters used for ESN initialization: *IS* (Input Scaling), *SR* (Spectral Radius) and α (Sparsity).

(1) *IS* is used for the initialization of the matrix $\mathbf{W}^{in}$: the elements of $\mathbf{W}^{in}$ are drawn from the uniform distribution in $[-IS, IS]$.

(2) *SR* is the spectral radius of $\mathbf{W}^{res}$, given by

$$\mathbf{W}^{res} = SR \cdot \frac{\mathbf{W}}{\lambda_{max}(\mathbf{W})} \tag{3}$$

where $\lambda_{max}(\mathbf{W})$ is the largest eigenvalue of matrix $\mathbf{W}$ and elements of $\mathbf{W}$ are generated randomly in $[-0.5, 0.5]$.

(3) α is the proportion of the non-zero elements in $\mathbf{W}^{res}$.

An ESN has two core properties for dynamical system identification.

(1) *Temporal Kernel*: Input time series drive the large reservoir and produce echo states in a high-dimensional state space, which enables reservoir to play a similar role as the kernel in kernel-based methods. That is, a reservoir can be regarded as a temporal kernel and the echo states are non-linear high-dimensional representations of the input time series.

(2) *Echo-State Property (ESP)* (Jaeger, 2001): The ESP means that inputs with similar short-term histories will evoke similar echo states, which ensures the dynamical stability of reservoir. The sufficient conditions for the ESP of standard sigmoid ESNs and leaky integrator ESNs are discussed in Yildiz, Jaeger, and Kiebel (2012). For standard sigmoid ESNs, the ESP is satisfied if recurrent weight matrix $\mathbf{W}^{res}$ is diagonally Schur stable. The ESP also provides ESNs with an important capability called "fading memory" or "short-term memory". With this short-term memory, the effect of the input and the previous reservoir state on future reservoir states should vanish gradually as time passes.

3. Related work

3.1. Improvement of ESN's output layer

ESNs have been successfully applied to many dynamic tasks, since they are able to model temporal dependencies, and the training of ESNs is efficient. However, the most common readout from the reservoir is a simple linear readout, which limits the decoding capability of ESNs (Boccato, Lopes, Attux, & Zuben, 2012; Boccato, Soriano, Attux, & Zuben, 2012; Lukoševičius, 2007). In the past decade, a large amount of research has focused on improving the output mapping of ESNs.

One approach is to improve the decoding capability by replacing the linear output layer with a multi-layer perceptron trained by backpropagation (Babinec & Pospíchal, 2006; Lukoševičius, 2007). Another approach is to use a random, non-linear projection of the input to the output layer, followed by a single-layer perceptron, an application of the idea originally proposed by Rosenblatt (Rosenblatt, 1958). This random projection idea has been followed up by later authors (Bin Huang, Yu Zhu, & Kheong Siew, 2004), and is now called Extreme Learning Machines (ELMs). This is the feed-forward analog of the ESN. One such model is the φ-ESN (Gallicchio & Micheli, 2011), which adds a fixed, random feed-forward layer (with tanh activations) after reservoir, followed by linear regression on the nonlinear projections. Hence this approach also provides a way to create a nonlinear decoder of the reservoir state, and therefore improves the non-linear separation capabilities of an ESN. In Butcher, Verstraeten, Schrauwen, Day, and Haycock (2010), a similar model named R^{2}SP was proposed by adding a fixed feedforward nonlinear projection layer after both the input layer and the reservoir.

In addition to the methods using ELM, Boccato et al. proposed replacing the output layer with a Volterra filter structure (Boccato et al., 2012, 2012). The Volterra filter not only enables better exploitation of the higher-order statistics of the input signals, but also preserves the simplicity of the training process.

The approaches mentioned above enhance the decoding capability of ESNs. Although these methods perform quite well on time series prediction tasks, they cannot be directly applied to time series classification tasks since they still cannot use the echo states of all time steps to determine the category. In contrast, EMN uses multi-scale convolution operations to decode the echo states at all time steps and max-over-time pooling to select the most discriminative features. Our state-of-the-art results demonstrate the effectiveness of the EMN.

3.2. ESN-based classifiers

Although ESNs have achieved excellent performance on time series prediction tasks thanks to the powerful temporal modeling capacity of reservoir, the problem of how to use the advantages of ESNs to deal with time series prediction tasks still needs exploration. In recent years, there has been some work that tries to employ an ESN in time series classification tasks. This work can be roughly divided into two categories. The first approach builds predictive models of each category, and classifies the data based on which model fits the data best. The second approach simply tries to categorize the data at each time step, and then averages the predictions over the time series.

Skowronski and Harris proposed a predictive ESN classifier and applied it to classification of spoken digits (Skowronski & Harris, 2007). A predictive ESN classifier is created for each digit and is trained to predict the input at next time step, and thus the test samples are classified as the model producing the lowest prediction error. Chen et al. proposed a model-metric co-learning (MMCL) methodology for time series classification tasks (Chen et al., 2015), which used a simplified ESN architecture with just three parameters, which were actually trained using Real-Time Recurrent Learning (Williams & Zipser, 1989). MMCL used separate output weight vectors for each sequence, trained as a predictor. The metric part of MMCL applied to these output weights, such that sequences from the same category were trained to have similar weights under a distance metric. In this way, classification uses nearest neighbor on the output weight vectors, after a new sequence is trained from the reservoir using ridge regression. However, these ESN-based methods are still predictive models rather than direct classifiers that can project temporal signals into discrete class labels, since they train the model by predicting the next item of a sequence.

In Verstraeten et al. (2007), the authors classify speech signals for 10 digits by constructing 10 one-vs-all classifiers. Each classifier is trained to output the correct class label at each time step and the final result at testing phase is obtained by taking the temporal mean of the output of every classifier. Similarly, Lukoševičius and Jaeger suggest a continuous readout of the category outputs, but weighting the end of the sequence

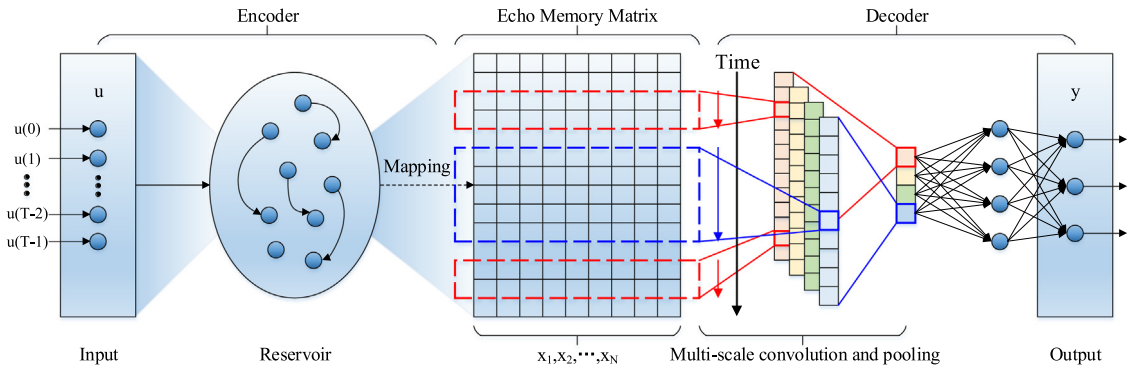


Fig. 2. General architecture of the Echo Memory Network. Here we assume a 1-dimensional input time series and 3 output categories. The red dashed box represents a convolution of scale 2 and the blue dashed box represents a convolution of scale 5 . (For interpretation of the references to color in this figure legend, the reader is referred to the web version of this article.)

more heavily, as enough information has reached the classifier. However, it is unclear that this is optimal — local variation in the signal may fit better with a different category. Our approach, in contrast, takes into account the information of the entire sequence, applying multi-scale convolutions on the unrolled echo state over time (which we call the echo memory matrix) to extract multi-scale discriminative temporal features, and using max-over-time pooling to select the most discriminative local patterns for classification.

4. Proposed approach

Combining the merits of ESNs (temporal kernel, ESP) and CNNs (multi-scale feature learning, temporal shift-invariance), we propose an end-to-end architecture called the Echo Memory Network (EMN) to deal with the time series classification task. The general structure is illustrated in Fig. 2. This architecture can be extended to deal with 3D skeleton-based action recognition tasks by using a multi-channel spatial information fusion strategy (described below).

4.1. Echo memory network

The EMN framework consists of two stages: an encoding and a decoding stage. In the encoding stage, each frame of the input time series is mapped into a high-dimensional reservoir state space and the echo state representations (ESRs) are extracted. The ESRs of all time steps are stored in a memory matrix. In the decoding stage, multi-scale convolutions and max-over-time pooling are used to decode the memory matrix. Then the pooled features are passed through a fully connected layer and the conditional probability distribution of the category is calculated by a softmax layer. We detail these parts below.

4.1.1. Echo memory encoder

The echo state is initialized to the zero vector. Assume a K-dimensional time series $\mathbf{u}$ (an example with K=1 is shown in Fig. 2) is presented to the model:

$$\mathbf{u} = (\mathbf{u}(0), \mathbf{u}(1), \ldots, \mathbf{u}(T-1))^T \tag{4}$$

At each time step, the echo state, $\mathbf{x}(t)$ is computed according to Eq. (1), and added to the Echo Memory Matrix (EMM) $\mathbf{X}$, resulting in:

$$\mathbf{X} = \begin{pmatrix} x_1(0) & x_2(0) & \ldots & x_N(0) \\ x_1(1) & x_2(1) & \ldots & x_N(1) \\ \vdots & \vdots & \vdots & \vdots \\ x_1(T-1) & x_2(T-1) & \ldots & x_N(T-1) \end{pmatrix} \tag{5}$$

at the end of the sequence. We use $\mathbf{x}(t) \in \mathbb{R}^N$ to denote the tth row of $\mathbf{X}$, the echo state at tth time step, $t \in 0, \ldots, T-1$, and $\mathbf{x}_j$ to denote the jth column of $\mathbf{X}$, the jth dimension of the echo states, $j \in 1, \ldots, N$. By collecting the echo states into the echo memory matrix, we have a complete representation of the sequence, and can then determine the category label by using a decoder to extract discriminative features from it.

4.1.2. Multi-scale convolutional decoder

After the extraction and collection of echo states into the EMM, a multi-scale convolution is used to extract local temporal information from it. We applied the convolution operations to the matrix along the direction of time and use multiple filters for each time scale.

As shown in Fig. 2, the convolutions are over the rows of $\mathbf{X}$, e.g., $\mathbf{x}(t), \mathbf{x}(t+1), \ldots \mathbf{x}(t+k-1)$, and so the convolution filters are of dimension $k \times N$. Fig. 2 shows two examples with $k = 2$ (red dashed lines) and $k = 4$ (blue dashed lines). Hence we consider k to be the time scale of the convolution. Each filter includes a bias term, and we use the standard ReLU activation function.

We use max-over-time pooling operation proposed in Collobert et al. (2011) to compute the maximum of each feature map. That is to say, we pool over the entire sequence and each filter results in one output, so the feature maps are reduced to one scalar each. Hence the pooling operation reduces the size of feature maps as well as the number of parameters for the next layer, which helps avoid overfitting and improves computational efficiency.

The max-over-time pooling operation introduces multi-scale temporal invariance. More importantly, it can be seen as a selection of the most important local patterns, no matter which time step the important patterns appear in. After the pooling procedure, all the pooled features are concatenated into a single vector and then fed into a fully connected layer followed by a softmax layer.

4.2. EMN with Spatial Information Fusion Strategy

When applying the proposed model to time series with spatial information such as 3D skeleton sequences, there are some details of the model that differ from dealing with general time series. For general multidimensional time series, there may be no spatial association between different dimensions. However, there is a spatial relationship between skeleton joints of different parts of the body. For example, there is a strong spatial relation between two legs when the human is running. Generally speaking, the relationship between the two arms and the relationship between two legs are highly correlated. Hence, the skeleton data

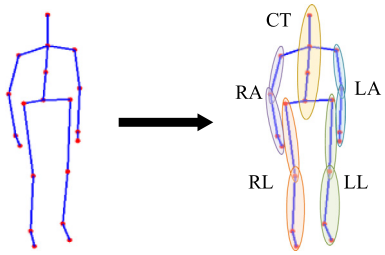


Fig. 3. Illustration of 3D skeleton joints being divided into 5 parts.

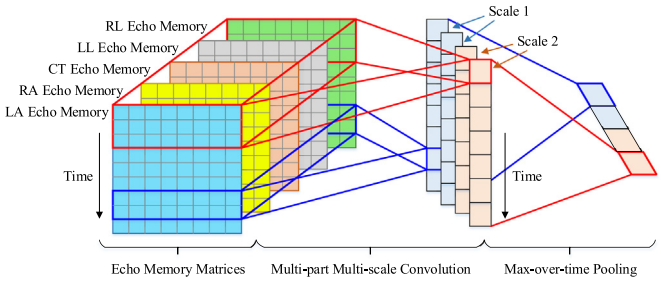
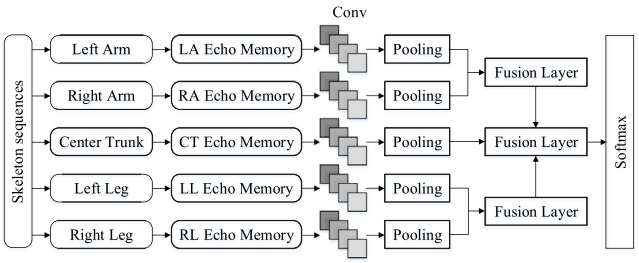
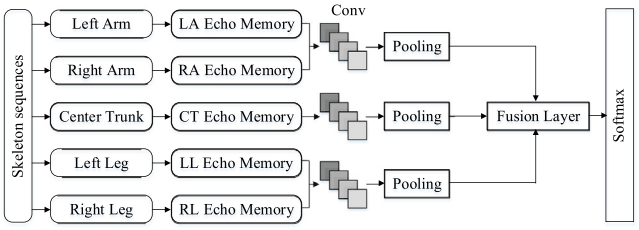


Fig. 4. Multi-part multi-scale convolutional process. Here is an example with 5 EMMs, 4 filters and 2 time scales.



(a) EMN-LSIF (Late Spatial Information Fusion)



(b) EMN-ESIF (Early Spatial Information Fusion)

Fig. 5. Two different multi-step fusion strategies of EMN.

of the two legs and the two arms should be combined before combining information from all the skeleton joints. In this subsection, we will introduce the EMN with two spatial information fusion strategies for dealing with 3D skeleton sequences.

4.2.1. Multi-part echo memory matrix representation

First, the human skeleton data is divided into five parts: left arm (LA), right arm (RA), left leg (LL), right leg (RL) and center trunk (CT). An example of 3D skeleton sequence being divided into five parts is shown in Fig. 3. Then the joint trajectories of the five parts are input into five separate reservoirs, obtaining five EMMs as described in the previous subsection. The echo memory matrix of each part is denoted as $\mathbf{X}^i$, where $\mathbf{X}^i$ is the ith channel of ESRs and $i \in 1, 2, \ldots, 5$.

4.2.2. Multi-part multi-scale convolution

In the convolution layer, multi-scale convolutions are used to extract multi-scale temporal features from the EMMs.

In order to fuse the information from different parts, the multi-part multi-scale convolution operations are applied to multiple EMMs. The multi-part multi-scale convolutions are illustrated in Fig. 4. Specifically, this is an example with 5 EMMs, 4 filters and 2 time scales. Depending on the strategies in next Section 4.2.3 for fusing the information from the five EMMs, the convolutions may be applied to just one of the EMMs, or may span both arms or both legs. The max-over-time pooling operation mentioned above is then used to select the important patterns.

4.2.3. Spatial information fusion strategy

In order to deal with the multivariate time series with spatial information, two strategies are used to capture the spatial correlation of the five skeleton parts, which we call spatial information fusion strategies. Specifically, we first fuse the features of two arms and two legs respectively and then we fuse all the features. These two strategies are introduced below.

The first strategy is the model called EMN with Late Spatial Information Fusion (EMN-LSIF). As shown in Fig. 5(a), five separate convolution layers and pooling layers are applied to the EMM of the five reservoirs respectively. After that, two fully-connected hidden layers are used, one for the concatenated features of the two arms, one for the two legs. These are integrated with the trunk output in a final hidden layer as in Fig. 5(a), followed by the softmax.

The second strategy is Early Spatial Information Fusion (EMN-ESIF). As shown in Fig. 5(b), the convolutions are over the two EMNs for the arms, and the two EMNs for the legs, resulting in convolutions of dimensions $k \times N \times 2$, as mentioned above. Hence the information from the arms and legs are fused first via the convolutions over both limbs. There are then only three convolutions, one over the arms, one over the legs, and a third over the trunk. The pooled features are then combined in a fully connected hidden layer and fed into the softmax to get the predicted conditional distribution.

In summary, the basic EMN architecture can be used for general time series classification tasks. When EMN is applied to a time series with spatial information, such as the action recognition task, we use the spatial information fusion strategies.

4.3. Training

The ADAM (Kingma & Ba, 2014) optimizer is used with learning rate of 0.001 to minimize the categorical cross-entropy loss:

$$L = -\sum_{i=1}^{n}\sum_{c=1}^{C} y_{i,c} \log \hat{y}_{i,c} \tag{6}$$

where $y_{i,c}$ is the target label of the ith sample of the cth category, and $\hat{\mathbf{y}}_{i,c}$ is the model output. C is the number of categories and n denotes the size of training set. To prevent overfitting, early stopping is used when the training loss flattens out. We again emphasize that the parameters of reservoir do not need to be learned.

5. Experiments

To evaluate the performance of the proposed model, experiments are performed on the UCR time series classification archive (Chen et al., 2015) to compare EMN with other methods. In addition, EMN is extended to deal with 3D skeleton-based action recognition benchmarks, as described above. We also gain more

Table 1
Description of 55 UCR time series classification datasets..

Dataset	#Classes	#Train	#Test	Length	Type
Adiac	37	390	391	176	Image Outline
Chlorine	3	467	3840	166	Simulated
Computers	2	250	250	720	Electric Device
CricketX	12	390	390	300	Motion Capture
CricketY	12	390	390	300	Motion Capture
CricketZ	12	390	390	300	Motion Capture
DistPhxAgeGp	3	139	400	80	Image Outline
DistPhxCorr	2	276	600	80	Image Outline
DistPhxTW	6	139	400	80	Image Outline
Earthquakes	2	139	322	512	Sensor Reading
ECG200	2	100	100	96	ECG
ECG5000	5	500	4500	140	ECG
ElectricDevices	7	8926	7711	96	Electric Device
FaceAll	14	560	1690	131	Image Outline
FacesUCR	14	200	2050	131	Image Outline
FiftyWords	50	450	455	270	Image Outline
Fish	7	175	175	463	Image Outline
FordA	2	1320	3601	500	Sensor Reading
FordB	2	810	3636	500	Sensor Reading
Ham	2	109	105	431	Spectrograph
HandOutlines	2	370	1000	2709	Image Outline
Haptics	5	155	308	1092	Motion Capture
InlineSkate	7	100	550	1882	Motion Capture
InsWngSnd	11	220	1980	256	Sensor Reading
LrgKitApp	3	375	375	720	Electric Device
MedicalImages	10	381	760	99	Image Outline
MidPhxAgeGp	3	154	400	80	Image Outline
MidPhxCorr	2	291	600	80	Image Outline
MidPhxTW	6	154	399	80	Image Outline
NonInv_Thor1	42	1800	1965	750	ECG
NonInv_Thor2	42	1800	1965	750	ECG
OSULeaf	6	200	242	427	Image Outline
PhalCorr	2	1800	858	80	Image Outline
Phoneme	39	214	1896	1024	Sensor Reading
Plane	7	105	105	144	Sensor Reading
ProxPhxAgeGp	3	400	205	80	Image Outline
ProxPhxCorr	2	600	291	80	Image Outline
ProxPhxTW	6	205	400	80	Image Outline
RefDev	3	375	375	720	Electric Device
ScreenType	3	375	375	720	Electric Device
ShapesAll	60	600	600	512	Image Outline
SmlKitApp	3	375	375	720	Electric Device
StarlightCurves	3	1000	8236	1024	Sensor Reading
Strawberry	2	370	613	235	Spectrograph
SwedishLeaf	15	500	625	128	Image Outline
Synth_Cntr	6	300	300	60	Simulated
Trace	4	100	100	275	Sensor Reading
TwoPatterns	4	1000	4000	128	Simulated
UWavGest_X	8	896	3582	315	Motion Capture
UWavGest_Y	8	896	3582	315	Motion Capture
UWavGest_Z	8	896	3582	315	Motion Capture
UWavGestAll	8	896	3582	945	Motion Capture
Wafer	2	1000	6174	152	Sensor Reading
WordSynonyms	25	267	638	270	Image Outline
Yoga	2	300	3000	426	Image Outline

insight into how the system works through visualization analysis and investigate the model's sensitivity to hyperparameter selection. These experiments are run on an Intel Core i7-6850K, 3.60-GHz CPU, 64-GB RAM and a GeForce GTX 1080-Ti 11G GPU.

5.1. UCR time series classification

5.1.1. Dataset introduction

The UCR time series classification archive (Chen et al., 2015) contains 85 publicly available time series datasets which vary by the number of classes, dataset types, number of samples, and length of time series. They are therefore useful for testing the general ability of a classifier over a variety of settings, however, we do not expect that one algorithm will be best on all of them. Each dataset was split into training and testing set by the provider. We chose 55 datasets with training set sizes of 100 or greater. Their characteristics are described in Table 1.

5.1.2. Evaluation metric

We evaluate the models based on their accuracy on each dataset, in addition, we obtain an overall error rate using the Mean Per-Class Error (MPCE), as proposed by Wang et al. (Wang, Yan, & Oates, 2017). MPCE is the expected error rate per class across all datasets, so it controls for the number of classes. Let c_k denote the number of classes in the kth dataset and e_k denote the error rate on the kth dataset, then the MPCE score of the model can be calculated as follows:

$$PCE_k = \frac{e_k}{c_k} \tag{7}$$

$$MPCE = \frac{1}{K}\sum_k PCE_k \tag{8}$$

where K denotes the number of datasets. By normalizing by the number of classes across datasets, an overall per-category error rate can be obtained.

5.1.3. Implementation details

In these experiments, the reservoir size N is fixed to 32 and SR to 0.9. The IS is chosen from $IS \in \{0.1, 1\}$, and the sparsity of reservoir is chosen from $\alpha \in \{0.3, 0.7\}$. For the multi-scale convolutional layer, two different time scales (i.e., the length of the temporal sliding window) are selected for each specific task according to the length of the time series. Specifically, for a T-length time series, we chose 2 neighboring time scales from $\{(0.1T, 0.2T), (0.2T, 0.3T), \ldots, (0.7T, 0.8T)\}$. If we find that even $(0.1T, 0.2T)$ leads to overfitting, we halve the time scales until we obtain a good result. The number of filters is chosen from $\{30, 60, 90, 120, 150\}$. The size of the fully connected layers is 64 or 128, and we do not use the fully connected layer for some datasets to mitigate overfitting. Dropout is applied to the input to softmax layer for some datasets to improve the generalization capability and the dropout rate is 0.25.

The hyper-parameter optimization process is divided into two stages to reduce the parameter space. First, the time scale is fixed at $(0.1T, 0.2T)$ and other parameters are estimated. Then other parameters are fixed and the time scale is tuned. For both stages, the hyper-parameters are estimated by a grid search via cross validation. More fine-grained parameter tuning for each dataset could potentially achieve better accuracy, but will result in a search over a very large parameter space. However, our results demonstrate that EMN achieves good performance even with this coarse tuning approach to our hyper-parameters. The hyper-parameters of EMN on each dataset are shown in detail in the supplementary material.

5.1.4. Comparison methods

For making a comprehensive evaluation, our model is first compared with traditional machine learning methods that have achieved good results on time series classification tasks. Then the proposed model is compared with several deep learning models.

For traditional machine learning methods, two classical yet powerful baseline methods are selected, 1-Nearest Neighbor with Euclidean distance (ED) and 1-Nearest Neighbor with Dynamic Time Warping (DTW) (Berndt & Clifford, 1994). We also compare with ten state-of-the-art methods published within the last five years. These methods can be roughly divided into three categories: distance-based methods, feature-based methods and ensemble-based methods. For fair comparison, we do not implement these methods but adopt the experimental results collected by Bagnall, Lines, Bostrom, Large, and Keogh (2017). These methods are briefly introduced below.

- **Distance-based Methods:** These methods measure the similarity of two given time series through predefined similarity measures, and then the classification can be done by using k-nearest neighbors (kNN). For the distance-based methods, in addition to the two classical methods, derivative DTW (DD_{DTW}) (Górecki & Łuczak, 2013) and derivative transform distance (DTD_C) (Górecki & Łuczak, 2014) are also selected. DD_{DTW} is an approach using a weighted combination of the DTW distance between two time series and one between their corresponding first-order difference series. Based on DD_{DTW}, DTD_C further considers the DTW distance between the sequences transformed by the sin, cosine and Hilbert transform.
- **Feature-based Methods:** These methods extract representative features from the raw time series to represent local or global patterns, and then classifies them based on these. EMN is compared with five feature-based methods. Learned shapelets (LS) (Grabocka, Schilling, Wistuba, & Schmidt-Thieme, 2014) is a classifier based on the shapelet transformation, and adopts a heuristic gradient-descent-based shapelet search procedure rather than enumeration. Bag of SFA symbols (BOSS) (Schäfer, 2015) uses windows to form "words" over the series and uses a truncated Discrete Fourier Transform on each window to obtain the features. Time series forest (TSF) (Deng, Runger, Tuv, & Vladimir, 2013) divides the time series into different intervals and calculates the mean, standard deviation and slope as the interval features, and then the intervals are randomly selected to train a forest of trees. Time series bag of features (TSBF) (Baydogan, Runger, & Tuv, 2013) selects multiple random length subsequences from random locations, and then partitions these subsequences into shorter intervals to capture local information. Learned pattern similarity (LPS) (Baydogan & Runger, 2016) is also based on intervals, but the main difference is that the subsequences themselves are used as attributes rather than the extracted interval features.
- **Ensemble-based Methods:** These methods combine different classifiers to achieve high performance. The performance of EMN is compared with three ensemble-based methods: Shapelet transform (ST) (Hills, Lines, Baranauskas, Mapp, & Bagnall, 2014), Elastic ensemble (EE) (Lines & Bagnall, 2015) and collection of transformation ensembles (COTE) (Bagnall, Lines, Hills, & Bostrom, 2015). ST uses the shapelet transformation to get the new representation of raw time series and construct a classifier from this representation using a weighted ensemble of 8 different classifiers. EE is the combination of 1-NN classifiers based on 11 elastic distance measures and uses a voting scheme to combine them. COTE uses weighted votes over 35 different classifiers, where the weights are proportional to their cross-validation accuracy on the training data.

For deep learning methods, effective methods have been proposed in recent years and applied to time series classification tasks. In Wang et al. (2017), Wang et al. applied three different deep neural network models to the UCR datasets, and good results were obtained. In order to make a comprehensive assessment of our model, EMN is also compared with these three models. These three methods are introduced below.

- **Multilayer Perceptrons (MLP):** The MLP model consists of three fully connected layers with 500 units per layer, and uses a softmax layer to get the final result. This model uses dropout and ReLU activation.
- **Fully Convolutional Networks (FCN):** The FCN model stacks three 1-D convolution blocks with 128, 256, and 128 filters in each block, with kernels of size 3, 5, and 8. After the convolution blocks, the features are fed into a global average pooling layer and a softmax layer to get the final result. The FCN model uses ReLU activation function and batch normalization.
- **Residual Network (ResNet):** Here, Wang, et al. stack three residual blocks comprised of three convolutional blocks each. The number of filters in the three residual blocks is 64, 128, and 128, respectively. This model also uses global average pooling layer and softmax layer.

Moreover, we also construct a baseline method named ConvLSTM to verify the effectiveness of our model, where an LSTM network replaces the ESN in our EMN. The number of hidden units is set to 32 to match the reservoir size of EMN and the same convolutional layer parameters as EMN are used in ConvLSTM.

5.2. UCR Time series classification results

In this subsection, the accuracy of EMN is first compared with the methods described above. Then the computational efficiency of EMN is compared to that of ConvLSTM. Finally, the results of ablation studies are presented to evaluate the contribution of the components of the model.

Comparison with Traditional Methods: The classification accuracies of EMN and other 12 strong time series classifiers on 55 UCR datasets are shown in Table 2. The average rank of each method as well as the number of datasets on which each model gets the best results are also summarized in Table 2.

As shown in Table 2, EMN achieves the best results on 19 of the 55 datasets amongst all the methods. EMN also achieves the highest average arithmetic rank of 2.955. Moreover, the MPCE value of EMN is 0.0403, the best of all models. However, there is no free lunch, and EMN is not the best on all datasets. For example, on TwoPatterns, even the simple 1NN-DTW classifier performs with higher accuracy. The simpler methods tend to perform well on datasets that nearly every method finds easy, i.e., with 100% performance. All in all, EMN performs well on a wide variety of datasets.

The next best performers are two ensemble-based methods, COTE, which combines the results from 35 different classifiers, and ST, the method based on the shapelet transform. These two methods are next best in all three metrics: the number of datasets they are best on, their average rank, and their MPCE.

To make a rigorous comparison, a non-parametric statistical test, the Nemenyi test (Demšar, 2006) is conducted on the average ranks of the classifiers. As shown in Fig. 6, the critical difference is 2.460, which means that two classifiers are not significantly different at the $p < 0.05$ level when their rank difference is less than 2.460. In this analysis, despite the numerical superiority of EMN, its performance was not significantly different from COTE and ST. However, these two methods are both ensemble-based. It is noteworthy that although ensemble-based methods achieve a higher accuracy than most single algorithm methods, they suffer from high computational complexity due to the sheer number of models included in their ensemble. In contrast, EMN is highly computationally efficient by using an untrained recurrent layer and only one convolutional layer. EMN is compared with these ensemble-models to show that the single EMN model can achieve equivalent performance even when compared to ensemble-based models.

Comparison with other Deep Learning Models: The accuracies of EMN and ConvLSTM as well as three deep learning models are shown in Table 3. EMN is better on 27 of the 55

Table 2
Accuracy of EMN and 12 strong time series classifiers on 55 UCR time series classification datasets..

Dataset	ED	DTW	DD_{DTW}	DTD_C	LS	BOSS	TSF	TSBF	LPS	ST(8)*	EE(11)*	COTE(35)*	EMN
Adiac	0.611	0.604	0.701	0.701	0.522	0.765	0.731	0.770	0.770	0.783	0.665	0.790	**0.829**
Chlorine	0.650	0.648	0.708	0.713	0.592	0.661	0.720	0.692	0.608	0.700	0.656	0.727	**0.845**
Computers	0.576	0.700	0.716	0.716	0.584	**0.756**	0.720	**0.756**	0.680	0.736	0.708	0.740	0.716
CricketX	0.577	0.754	0.754	0.754	0.741	0.736	0.664	0.705	0.697	0.772	**0.813**	0.808	0.782
CricketY	0.567	0.744	0.777	0.774	0.718	0.754	0.672	0.736	0.767	0.779	0.805	**0.826**	0.787
CricketZ	0.587	0.754	0.774	0.774	0.741	0.746	0.672	0.715	0.754	0.787	0.782	**0.815**	0.808
DistPhxAgeGp	0.626	0.770	0.705	0.662	0.719	0.748	0.748	0.712	0.669	0.770	0.691	0.748	**0.843**
DistPhxCorr	0.717	0.717	0.732	0.725	0.779	0.728	0.772	0.783	0.721	0.775	0.728	0.761	**0.822**
DistPhxTW	0.633	0.590	0.612	0.576	0.626	0.676	0.669	0.676	0.568	0.662	0.647	0.698	**0.795**
Earthquakes	0.712	0.719	0.705	0.705	0.741	0.748	0.748	0.748	0.640	0.741	0.741	0.748	**0.811**
ECG200	0.880	0.770	0.830	0.840	0.880	0.870	0.870	0.840	0.860	0.830	0.880	0.880	**0.920**
ECG5000	0.925	0.924	0.924	0.924	0.932	0.941	0.939	0.940	0.917	0.944	0.939	**0.946**	0.944
ElectricDevices	0.552	0.602	0.592	0.594	0.587	**0.799**	0.693	0.703	0.681	0.747	0.663	0.713	0.716
FaceAll	0.714	0.808	0.902	0.899	0.749	0.782	0.751	0.744	0.767	0.779	0.849	**0.918**	0.903
FacesUCR	0.769	0.905	0.904	0.908	0.939	**0.957**	0.883	0.867	0.926	0.906	0.945	0.942	0.947
FiftyWords	0.631	0.690	0.754	0.754	0.730	0.705	0.741	0.758	0.818	0.705	**0.820**	0.798	0.758
Fish	0.783	0.823	0.943	0.926	0.960	**0.989**	0.794	0.834	0.943	0.989	0.966	0.983	0.960
FordA	0.665	0.555	0.723	0.765	0.957	0.930	0.815	0.850	0.873	**0.971**	0.738	0.957	0.932
FordB	0.606	0.620	0.667	0.653	**0.917**	0.711	0.688	0.599	0.711	0.807	0.662	0.804	0.908
Ham	0.600	0.467	0.476	0.552	0.667	0.667	0.743	0.762	0.562	0.686	0.571	0.648	**0.781**
HandOutlines	0.862	0.881	0.868	0.865	0.481	0.903	0.919	0.854	0.881	**0.932**	0.889	0.919	0.891
Haptics	0.370	0.377	0.399	0.399	0.468	0.461	0.445	0.490	0.432	**0.523**	0.393	**0.523**	0.519
InlineSkate	0.342	0.384	**0.562**	0.509	0.438	0.516	0.376	0.385	0.500	0.373	0.460	0.495	0.460
InsWngSnd	0.562	0.355	0.355	0.473	0.606	0.523	0.633	0.625	0.551	0.627	0.595	**0.653**	0.641
LrgKitApp	0.493	0.795	0.795	0.795	0.701	0.765	0.571	0.528	0.717	0.859	0.811	0.845	**0.901**
MedicalImages	0.684	0.737	0.737	0.745	0.664	0.718	0.755	0.705	0.746	0.670	0.742	0.758	**0.775**
MidPhxAgeGp	0.519	0.500	0.539	0.500	0.571	0.545	0.578	0.578	0.487	0.643	0.558	0.636	**0.800**
MidPhxCorr	0.766	0.698	0.732	0.742	0.780	0.780	**0.828**	0.814	0.773	0.794	0.784	0.804	0.815
MidPhxTW	0.513	0.506	0.487	0.500	0.506	0.545	0.565	0.597	0.526	0.519	0.513	0.571	**0.639**
NonInv_Thor1	0.829	0.790	0.806	0.841	0.259	0.838	0.876	0.842	0.812	**0.950**	0.846	0.931	0.933
NonInv_Thor2	0.880	0.865	0.893	0.890	0.770	0.901	0.910	0.862	0.841	**0.951**	0.913	0.946	0.939
OSULeaf	0.521	0.591	0.880	0.884	0.777	0.955	0.583	0.760	0.740	**0.967**	0.806	**0.967**	0.897
PhalCorr	0.761	0.728	0.739	0.761	0.765	0.772	0.803	0.830	0.756	0.763	0.773	0.770	**0.832**
Phoneme	0.109	0.228	0.269	0.268	0.218	0.265	0.212	0.276	0.237	0.321	0.305	**0.349**	0.239
Plane	0.962	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ProxPhxAgeGp	0.785	0.805	0.800	0.795	0.834	0.834	0.849	0.849	0.795	0.844	0.805	**0.854**	**0.854**
ProxPhxCorr	0.808	0.784	0.794	0.794	0.849	0.849	0.828	0.873	0.842	0.883	0.808	0.869	**0.890**
ProxPhxTW	0.707	0.761	0.766	0.771	0.776	0.800	0.815	0.810	0.732	0.805	0.766	0.780	**0.830**
RefDev	0.395	0.464	0.445	0.445	0.515	0.499	**0.589**	0.472	0.459	0.581	0.437	0.547	0.560
ScreenType	0.360	0.397	0.429	0.437	0.429	0.464	0.456	0.509	0.416	0.520	0.445	0.547	**0.555**
ShapesAll	0.752	0.768	0.850	0.838	0.768	**0.908**	0.792	0.185	0.873	0.842	0.867	0.892	0.873
SmlKitApp	0.344	0.643	0.640	0.648	0.664	0.725	**0.811**	0.672	0.712	0.792	0.696	0.776	0.699
StarlightCurves	0.849	0.907	0.962	0.962	0.947	0.978	0.969	0.977	0.963	0.979	0.926	**0.980**	0.978
Strawberry	0.946	0.941	0.954	0.957	0.911	**0.976**	0.965	0.954	0.962	0.962	0.946	0.951	0.971
SwedishLeaf	0.789	0.792	0.901	0.896	0.907	0.922	0.914	0.915	0.920	0.928	0.915	**0.955**	0.941
Synth_Cntr	0.880	0.993	0.993	0.997	0.997	0.967	0.987	0.993	0.980	0.983	0.990	**1.000**	0.997
Trace	0.760	**1.000**	**1.000**	0.990	**1.000**	**1.000**	0.990	0.980	0.980	**1.000**	0.990	**1.000**	**1.000**
TwoPatterns	0.907	**1.000**	**1.000**	**1.000**	0.993	0.993	0.991	0.976	0.982	0.955	**1.000**	**1.000**	0.999
UWavGest_X	0.739	0.728	0.779	0.775	0.791	0.762	0.804	**0.831**	0.829	0.803	0.805	0.822	0.813
UWavGest_Y	0.662	0.634	0.716	0.698	0.703	0.685	0.727	0.736	**0.761**	0.730	0.726	0.759	0.736
UWavGest_Z	0.650	0.658	0.696	0.679	0.747	0.695	0.743	**0.772**	0.768	0.748	0.724	0.750	0.755
UWavGestAll	0.948	0.892	0.935	0.938	0.953	0.939	0.957	0.926	0.966	0.942	**0.968**	0.964	0.958
Wafer	0.995	0.980	0.980	0.993	0.996	0.995	0.996	0.995	0.997	**1.000**	0.997	**1.000**	0.998
WordSynonyms	0.618	0.649	0.730	0.730	0.607	0.638	0.647	0.688	0.755	0.571	**0.779**	0.757	0.663
Yoga	0.830	0.837	0.856	0.856	0.834	**0.918**	0.859	0.819	0.869	0.818	0.879	0.877	0.866
Best	0	3	4	2	3	9	4	4	2	9	6	16	**19**
AVG rank	11.400	10.091	8.409	8.409	8.191	6.009	6.491	6.809	7.809	5.000	6.418	3.009	**2.955**
MPCE	0.0755	0.0690	0.0626	0.0620	0.0603	0.0525	0.0548	0.0566	0.0611	0.0486	0.0589	0.0469	**0.0403**

*ST is an ensemble of 8 classifiers, EE consists of 11 1-NN classifiers and COTE is an ensemble model with 35 different base models. EMN is compared with these ensemble-based methods to show that as a single model, EMN can achieve better performance even compared with ensembles.

datasets among all the methods for comparison, and achieves the highest average rank of 1.964 and the lowest MPCE value of 0.0403. We also conduct the Nemenyi test on the average ranks of the classifiers here. As shown in Fig. 7, the critical difference is 0.822 and EMN is significantly better than all the comparison methods except FCN. Although FCN also achieves a comparable result, it is worth noting that deep neural networks use multiple hidden layers while EMN uses only one untrained recurrent layer, a convolutional layer, and occasionally a hidden layer (see supplememntary materials). So the EMN has higher training efficiency and fewer training parameters.

The results in Table 3 also show that EMN obtains higher classification accuracy than ConvLSTM for the vast majority of datasets. More importantly, the training of ConvLSTM is time-consuming, especially for long time series.

Computational Efficiency: The training parameters and training time per epoch of EMN are compared with ConvLSTM on 18 UCR datasets. The results and the speed-up rates of our model are shown in Table 4. As can be seen from the Table, the training time of EMN is an order of magnitude faster than ConvLSTM, with a minimum speed-up rate of over 21 times. This result shows that EMN achieves excellent performance in TSC tasks while maintaining training efficiency. The automatic production of ESRs

Table 3
Accuracy of EMN and 4 deep learning models on 55 UCR datasets..

Dataset	MLP	FCN	Res_Net	ConvLSTM	EMN
Adiac	0.752	**0.857**	0.826	0.737	0.829
Chlorine	**0.872**	0.843	0.828	0.798	0.845
Computers	0.540	**0.848**	0.824	0.736	0.716
CricketX	0.569	0.815	**0.821**	0.705	0.782
CricketY	0.595	0.792	**0.805**	0.733	0.787
CricketZ	0.592	**0.813**	**0.813**	0.772	0.808
DistPhxAgeGp	0.827	0.835	0.798	0.838	**0.843**
DistPhxCorr	0.810	0.812	0.820	0.798	**0.822**
DistPhxTW	0.747	0.790	0.740	0.763	**0.795**
Earthquakes	0.792	0.801	0.786	**0.832**	0.811
ECG200	**0.920**	0.900	0.870	0.890	**0.920**
ECG5000	0.935	0.941	0.931	0.943	**0.944**
ElectricDevices	0.580	0.723	0.728	**0.765**	0.716
FaceAll	0.885	**0.929**	0.834	0.726	0.903
FacesUCR	0.815	0.948	**0.958**	0.920	0.947
FiftyWords	0.712	0.679	0.727	0.688	**0.758**
Fish	0.874	0.971	**0.989**	0.943	0.960
FordA	0.769	0.906	0.928	0.924	**0.932**
FordB	0.629	0.883	0.900	0.895	**0.908**
Ham	0.714	0.762	0.781	**0.790**	0.781
HandOutlines	0.807	0.776	0.861	0.865	**0.891**
Haptics	0.461	**0.551**	0.506	0.510	0.519
InlineSkate	0.351	0.411	0.365	0.420	**0.460**
InsWngSnd	0.631	0.402	0.531	0.635	**0.641**
LrgKitApp	0.480	0.896	0.893	0.867	**0.901**
MedicalImages	0.729	**0.792**	0.772	0.746	0.775
MidPhxAgeGp	0.735	0.768	0.760	**0.805**	0.800
MidPhxCorr	0.760	0.795	0.793	0.742	**0.815**
MidPhxTW	0.609	0.612	0.607	0.632	**0.639**
NonInv_Thor1	0.942	**0.961**	0.948	0.894	0.933
NonInv_Thor2	0.943	**0.955**	0.951	0.909	0.939
OSULeaf	0.570	**0.988**	0.979	0.690	0.897
PhalCorr	0.830	0.826	0.825	0.809	**0.832**
Phoneme	0.098	**0.345**	0.324	0.228	0.239
Plane	0.981	**1.000**	**1.000**	0.933	**1.000**
ProxPhxAgeGp	0.824	0.849	0.849	0.849	**0.854**
ProxPhxCorr	0.887	0.900	**0.918**	0.887	0.890
ProxPhxTW	0.797	0.810	0.807	0.818	**0.830**
RefDev	0.371	0.533	0.528	0.501	**0.560**
ScreenType	0.408	0.667	**0.707**	0.496	0.555
ShapesAll	0.775	0.898	**0.912**	0.770	0.873
SmlKitApp	0.389	**0.803**	0.797	0.691	0.699
StarlightCurves	0.957	0.967	0.975	0.966	**0.978**
Strawberry	0.967	0.969	0.958	0.967	**0.971**
SwedishLeaf	0.893	**0.966**	0.958	0.877	0.941
Synth_Cntr	0.950	0.990	**1.000**	0.983	0.997
Trace	0.820	**1.000**	**1.000**	**1.000**	**1.000**
TwoPatterns	0.886	0.897	**1.000**	**1.000**	0.999
UWavGest_X	0.768	0.754	0.787	0.787	**0.813**
UWavGest_Y	0.703	0.725	0.668	0.730	**0.736**
UWavGest_Z	0.705	0.729	0.755	**0.756**	0.755
UWavGestAll	0.954	0.826	0.868	0.934	**0.958**
Wafer	0.996	0.997	0.997	**0.999**	0.998
WordSynonyms	0.594	0.580	0.632	0.618	**0.663**
Yoga	0.855	0.845	0.858	0.822	**0.866**
Best	2	15	12	8	**27**
AVG rank	4.227	2.664	2.791	3.355	**1.964**
MPCE	0.0612	0.0419	0.0411	0.0452	**0.0403**

without training is effective in modeling temporal dependencies. The training via back-propagation through time of the LSTM network is clearly very time-consuming.

Ablation Studies: To demonstrate the effectiveness of the echo memory encoding and multi-scale convolutional decoding, EMN is compared with ablations of itself: (1) a multi-scale CNN with max-over-time pooling; (2) an ESN using mean-pooling echo state alone with fully connected layer and softmax layer (here we denote it as mean-ESN); and (3) an ESN-based classifier (Lukoševičius & Jaeger, 2009) that predicts the class label at each time step and conducts temporal mean pooling to obtain the final class probabilities (here we denote it as pred-ESN). For multi-scale CNN, the same time scales and number of filters as EMN

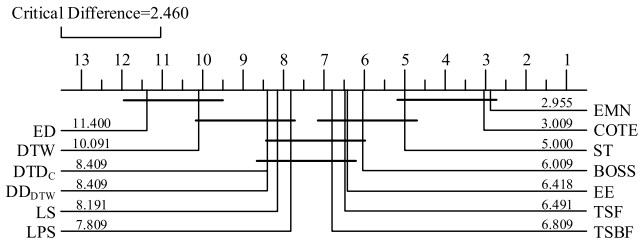


Fig. 6. Critical difference diagram over the average arithmetic rank of EMN and 12 strong time series classifiers. The methods connected in one group are not significantly different at 0.05 significance level.

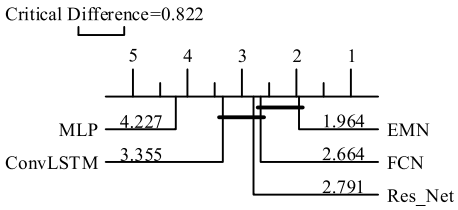


Fig. 7. Critical difference diagram over the average arithmetic rank of EMN and 4 deep learning models.

are used. For mean-ESN and pred-ESN, the same reservoir size as EMN is used, and linear regression is used to train pred-ESN. The classification accuracy comparison between EMN and its ablations are shown in Fig. 8.

In Fig. 8, EMN obtains higher accuracy in the vast majority of datasets compared with the multi-scale CNN since the echo memory matrix (EMM) contains abundant temporal features of the raw time series. On the other hand, the comparison with pred-ESN and mean-ESN shows that the multi-scale convolution and max-over-time poling is a powerful decoder for the EMM.

5.3. 3D Skeleton-based action recognition

Human action recognition is an active research branch in machine learning, with a wide range of applications. 3D skeleton joint data are the 3D coordinate trajectories of the joints in the skeleton, which can be used to identify human actions. Fig. 9 shows an example of a 3D skeleton sequence. In our experiments, the 3D skeleton sequences are regarded as a multivariate time series and the EMN model is used for the 3D skeleton-based action recognition task as described previously.

5.3.1. Dataset introduction

EMN is evaluated on four 3D skeleton-based action recognition benchmarks. The characteristics of these four datasets are described below.

MSR-Action 3D (MSRA3D) (Li, Zhang, & Liu, 2010): This dataset provides skeleton (20 joints) 3D coordinates data for 20 actions performed 2–3 times by 10 subjects, and gives a total of 567 sequences with 23,797 frames. In Zhang, Wang, Gou, Sznaier, and Camps (2016), 10 sequences with excessive noise were removed. However, we use the complete original datasets in order to evaluate the anti-noise capacity of our model.

HDM05 (Müller et al., 2007): An optical marker-based motion captures dataset sampled at 120 Hz. It contains 3D coordinates of 31 joints for 130 actions performed by 5 non-professional actors. Following the same protocol in Cho and Chen (2014), we classify some similar actions into the same category and obtain 65 action categories.

Florence3D-Action (Seidenari, Varano, Berretti, Bimbo, & Pala, 2013): This dataset contains 3D locations of 15 joints captured

Table 4
The comparison of training time (seconds) between EMN and ConvLSTM on 18 UCR datasets..

Dataset	Parameters		Training time		Speed-up rate
	ConvLSTM	EMN	ConvLSTM	EMN	
Computers	3.05E+04	2.61E+04	2.03E+01	2.53E−01	80.34×
Earthquakes	2.93E+05	2.89E+05	1.04E+01	2.45E−01	42.41×
ElectricDevices	5.94E+04	5.50E+04	2.97E+01	1.14E+00	26.08×
Fish	9.04E+05	8.99E+05	6.52E+00	2.48E−01	26.26×
Ham	6.42E+04	5.99E+04	5.85E+00	1.49E−01	39.33×
HandOutlines	1.30E+06	1.30E+06	1.22E+02	4.84E+00	25.25×
LrgKitApp	4.51E+05	4.46E+05	2.42E+01	6.63E−01	36.45×
MedicalImages	7.89E+04	7.45E+04	5.06E+00	2.31E−01	21.93×
NonInv_Thor1	6.60E+05	6.56E+05	5.52E+01	1.89E+00	29.18×
NonInv_Thor2	6.60E+05	6.56E+05	5.21E+01	1.92E+00	27.07×
OSULeaf	2.48E+05	2.44E+05	6.52E+00	2.13E−01	30.56×
PhalCorr	1.20E+05	1.16E+05	8.13E+00	3.06E−01	26.55×
Phoneme	1.18E+05	1.14E+05	5.34E+01	1.63E+00	32.76×
RefDev	1.06E+05	1.02E+05	2.33E+01	6.63E−01	35.09×
ScreenType	8.57E+04	8.14E+04	3.77E+01	8.78E−01	42.87×
SmlKitApp	3.03E+05	2.98E+05	3.36E+01	1.16E+00	29.05×
Trace	2.39E+05	2.34E+05	5.67E+00	1.51E−01	37.52×
WordSynonyms	7.36E+05	7.32E+05	3.42E+01	1.56E+00	21.88×

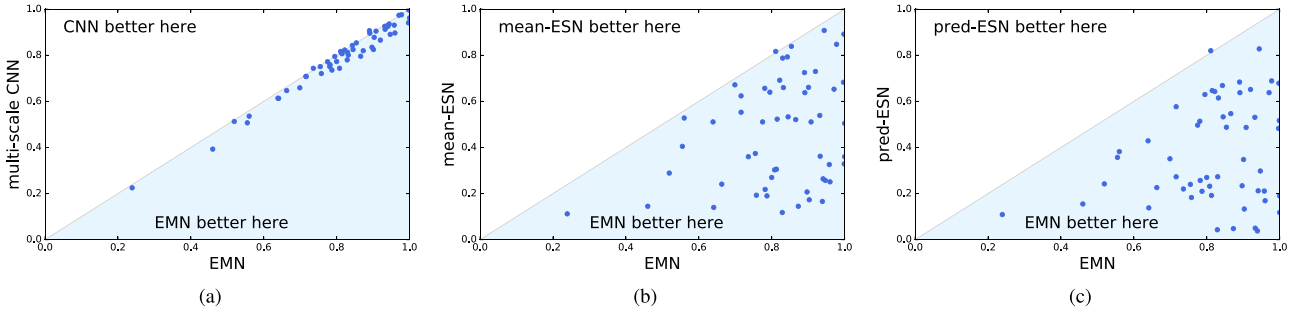


Fig. 8. Pairwise accuracy plots on 55 UCR datasets between EMN and (a) multi-scale CNN (b) mean-ESN that uses mean-pooling echo state alone with fully connected layer and softmax layer (c) pred-ESN that predicts the class label at each time step and conducts temporal mean pooling to obtain the final class label.

with a Kinect, for 9 actions performed 2–3 times by 10 different subjects, giving a total of 215 sequences. Its challenge is the high intra-class variations (left-hand actors and right-hand ones) and the presence of similar actions such as drinking from a bottle and answering the phone.

UTKinect-Action (UTKA) (Xia, Chen, & Aggarwal, 2012): This dataset provides 20-joint skeleton coordinates for 10 actions performed 2 times by 10 subjects, captured at 15FPS. There are 195 sequences. The frame length ranges from 5 to 170 with an average value of 30.5 ± 20. The challenge of this dataset is the intra-class variations and the multiple views.

5.3.2. Implementation details

In these experiments, the *IS* is set to 0.1, the sparsity α is 0.01, the *SR* is 0.99 and the reservoir sizes range from 100 to 300. For the multi-scale convolutional layer, we chose the length of the sliding windows to be 2, 3, & 4 and the number of filters for each length ranges from 16 to 128. The size of the final fusion layer is set as 144. These parameters are selected in a manner similar to the previous experiments. And the parameters on each datasets are shown in supplemental material in detail.

The skeleton joint sequences are preprocessed before training, since the raw skeleton joints are not in a unified coordinate system, and different joint trajectories have different levels of smoothness. The coordinate system is normalized by setting the origin to the average of the hip center, left, and right joints. Then the popular Savitzky–Golay smoothing filter (Steinier, Termonia, & Deltour, 1972) is applied to smooth the trajectories. To reduce computational cost on the HDM05 dataset, we down-sample every 4 frames. We do not down-sample the other datasets. Finally,

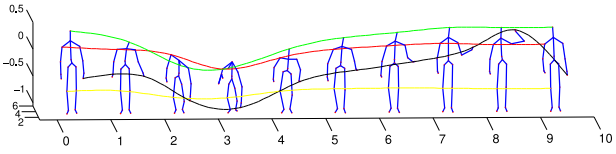


Fig. 9. An example of 3D skeleton sequence.

the variable length trajectories are padded with zeros up to a given max-length value.

5.3.3. Comparison methods

EMN is compared with some of the 3D skeleton-based action recognition methods proposed in recent years. These methods can be roughly divided into 3 categories: feature-based methods, dynamics-based methods and deep learning methods.

Feature-based methods extract discriminative pose representations and then perform action recognition. For the feature-based methods, our model is compared with eight methods: Covariance (Hussein et al., 2013), Skeletons Lie group (Vemulapalli et al., 2014), Gram matrices Representations (Zhang et al., 2016), Random Forest+depth (Zhu et al., 2013), SGWT+SVM (Kerola et al., 2014), DMMs+Fisher vectors (Chen et al., 2016), Multi-Part Bag-of-Poses (Seidenari et al., 2013) and DP+KNN (Devanne et al., 2013).

Dynamics-based methods treat skeleton data as 3D trajectories of joints, and human action recognition can be seen as a multivariate time series classification task. For the dynamics-based methods, EMN is compared with four methods: HOD (Gowayyed

Table 5
Recognition accuracy (%) on MSR-Action 3D dataset (Cross-subject Test).

Methods	Accuracy(%)
Covariance (Hussein, Torki, Gowayyed, & El-Saban, 2013)	88.10
HOD (Gowayyed, Torki, Hussein, & El-Saban, 2013)	91.26
Skeletons Lie group (Vemulapalli, Arrate, & Chellappa, 2014)	92.46
DHMM+SL (Lo Presti, La Cascia, Sclaroff, & Camps, 2015)	92.91
Random Forest+depth (Zhu, Chen, & Guo, 2013)	94.30
Hierarchical LSTM (Du, Wang, & Wang, 2015)	94.49
SGWT+SVM (Kerola, Inoue, & Shinoda, 2014)	94.77
DMMs+Fisher vectors (Chen et al., 2016)	95.97
Gram matrices Representations (Zhang et al., 2016)	96.97
EMN-LSIF	97.56
EMN-ESIF	**97.88**

Table 6
Recognition accuracy on HDM05 dataset (10-fold cross validation).

Methods	Accuracy(%)
DNN (Cho & Chen, 2014)	95.59
Hierarchical LSTM (Du et al., 2015)	96.92
Deep LSTM (Zhu et al., 2016)	**97.25**
EMN-LSIF	97.08
EMN-ESIF	**97.25**

Table 7
Recognition accuracy on Florence3D-Action dataset (10-fold cross validation).

Methods	Accuracy(%)
Multi-Part Bag-of-Poses (Seidenari et al., 2013)	82.00
Skeletons Lie group (Vemulapalli et al., 2014)	90.88
EMN-LSIF	91.17
EMN-ESIF	**91.72**

Table 8
Recognition accuracy on UTKinect Action dataset (10-fold cross validation).

Methods	Accuracy(%)
Random Forest+depth (Zhu et al., 2013)	87.90
LTBSVM (Slama, Wannous, Daoudi, & Srivastava, 2015)	88.50
HOJ3D+HMM (Xia et al., 2012)	90.92
DP+KNN (Devanne et al., 2013)	91.50
Skeletons Lie group (Vemulapalli et al., 2014)	97.08
Gram matrices Representations (Zhang et al., 2016)	**100.00**
EMN-LSIF	**100.00**
EMN-ESIF	**100.00**

et al., 2013), DHMM+SL (Lo Presti et al., 2015), LTBSVM (Slama et al., 2015), HOJ3D+HMM (Xia et al., 2012).

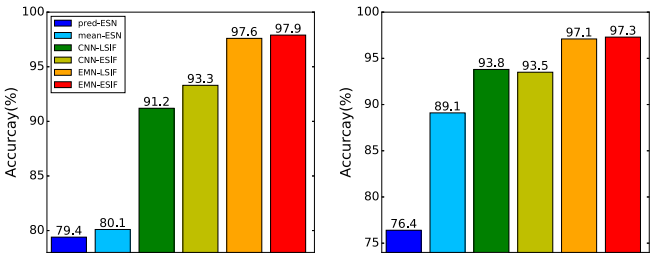
Deep learning methods use deep neural networks to deal with the 3D skeleton-based action recognition task. For deep learning methods, EMN is compared with three deep learning models: DNN (Cho & Chen, 2014), Hierarchical LSTM (Du et al., 2015) and Deep LSTM (Zhu et al., 2016).

5.4. Action recognition results

The experimental results on four 3D skeleton joint datasets will be illustrated below.

MSR-Action 3D: We use a standard validation protocol used by Li et al. (2010) on the MSR-Action 3D dataset. In this protocol, the whole dataset is split into three overlapping subsets (AS1, AS2, AS3) of 8 classes for each one. Within each set, we adopt cross-subject validation: the subjects 1, 3, 5, 7, 9 are used for training and 2, 4, 6, 8, 10 are used for testing. The results (average accuracy of AS1, AS2 and AS3) are reported in Table 5.

As seen from Table 5, EMN-ESIF achieves the best average accuracy with 97.88%, while EMN-LSIF also performs well with 97.56%. Without removing the 10 excessively noisy sequences,



(a) MSR-Action 3D (b) HDM05

Fig. 10. Accuracies(%) of EMN and its ablation models on (a) MSR-Action 3D and (b) HDM05.

they both outperform the existing approaches listed in Table 5. The best of the other methods is the work of Zhang et al. (Zhang et al., 2016) with 96.97%, where 10 noisy sequences were removed.

HDM05: Following the protocol used in Du et al. (2015), 10-fold cross validation is performed on this dataset. As shown in Table 6, EMN-ESIF and Deep LSTM (Zhu et al., 2016) achieve the best average accuracy with 97.25%.

Florence3D-Action: As shown in Table 7, EMN-LSIF and EMN-ESIF both outperform the approaches listed in Table 7.

UTKinect-Action: Table 8 shows that our models achieve the best performance of 100%, which is also reached by Zhang et al. (2016).

It can be seen from the above experimental results on four 3D skeleton sequence datasets that EMN-ESIF model always performs better than EMN-LSIF (except when both achieve 100% accuracy). In a late fusion strategy, the features are extracted along independent pathways rather than jointly, which reduces the information that can be shared. Therefore, the early fusion strategy works better here.

Ablation Studies: Similar to the previous experiments, EMN is compared with its ablation models: (1) pred-ESN and (2) mean-ESN as introduced in last subsection; (3) multi-scale CNN alone with LSIF; (4) multi-scale CNN alone with ESIF, to demonstrate the importance of the different components.

Fig. 10 shows that both EMN-LSIF and EMN-ESIF achieve higher accuracy than the multi-scale CNN baselines without an action echo memory (CNN-LSIF and CNN-ESIF). Aside from that, the ESN baselines without convolution decoder are poor classifiers compared to EMN. Clearly, using a CNN gives a big boost over the ESN-alone-based methods. Otherwise, the full EMN model is superior to all of its ablations, demonstrating the effectiveness of ESN's temporal feature encoding capability and CNN's multi-scale feature decoding capability. Since the sequences of the HDM05 dataset are much longer than the sequences of the MSRA3D dataset, and pred-ESN judges the action label at each time step, pred-ESN is comparable to mean-ESN on the shorter-sequence MSRA3D dataset but performs poorly on the HDM05 dataset.

5.5. Visualization analysis

In order to explore the interpretability of the EMN model, the echo memory matrix (EMM) of 3D skeleton sequences on the MSR-Action 3D dataset is visualized and their characteristics are analyzed. There are two illustrative examples given in Fig. 11 showing the human action and the corresponding EMM. Fig. 11(a) shows the skeleton joint trajectories of a person waving both hands and the heat map of the left arm echo memory. It can be seen that the pose at the 15th frame and the pose at the 35th frame are very similar, but the corresponding echo

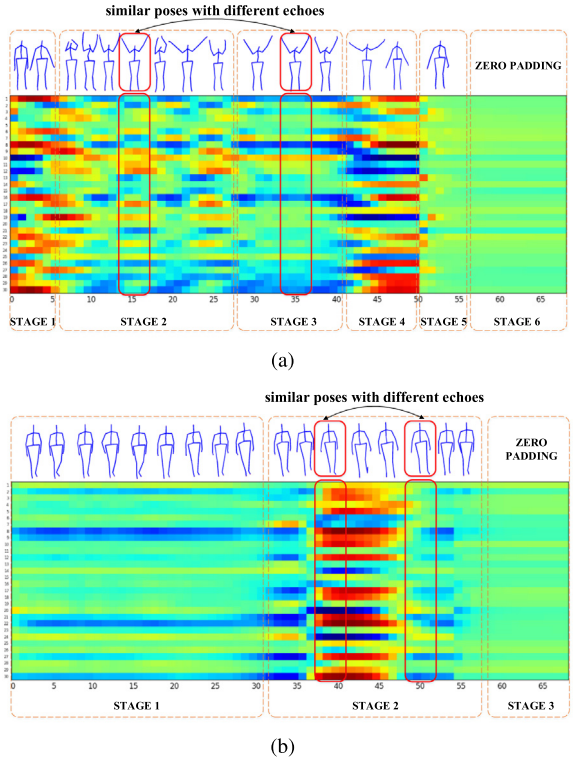


Fig. 11. The skeleton joint trajectories of a person (a) waving his hands and (b) kicking forward, and the heat map of corresponding echo states. The echo state representation of the same pose differs depending on the context.

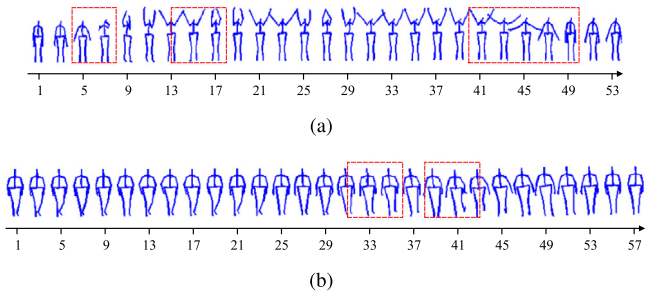


Fig. 12. The skeleton joint trajectories of a person (a) waving his hands and (b) kicking forward, and the important local patterns that are kept after max-over-time pooling.

states are quite different. The same phenomenon can also be found in Fig. 11(b), where similar poses with different preceding pose sequences (kicking forward and swinging back) have totally different echo states.

Fig. 11 can be analyzed from two perspectives. First, in the horizontal direction (time), the effect of contextual information can be seen in the representation of similar sub-actions with different histories, which makes it possible to classify different actions that "pass through" similar poses. Second, due to the high-dimensional projection and highly sparse connectivity of neurons in the reservoir, it can be seen in the vertical direction that the ESRs are varied and thus helpful in distinguishing actions.

Fig. 12 shows the skeleton joint trajectories and the important local patterns that are kept after pooling. Here we use 64 filters with a window length of 4 to convolve the EMM (we convolve the echo memory of two arms in Fig. 12(a) and the echo memory of two legs in Fig. 12(b)), and then record the position corresponding to the maximum value of each feature map. The position recorded by more than two filters is considered as an important local pattern. We use red dotted boxes with a length of 4 to mark these important local patterns and use a box with a length larger than 4 if multiple local patterns have overlapping parts. It can be seen from Fig. 12(a) that for recognizing the action of waving two hands, the important sub-actions are raising the hands, waving and putting the hands down. In Fig. 12(b), for recognizing the actions of kicking forward, the important sub-actions are lifting the foot backward and kicking the foot forward.

EMN achieves better performance on time series classification tasks since it can capture a large variety of temporal information from the input time series by using a reservoir as a temporal kernel, convolutions to extract temporal features and max-over-time pooling to select the most discriminative local patterns. In contrast, mean-ESN predicts the category label at each time step and some unimportant patterns (e.g., the static standing posture at the beginning of the forward kicking action in Fig. 12(b)) will interfere with the classification performance.

5.6. Sensitivity to hyperparameters

The hyperparameters affect the capability of reservoir and thus influence the performance of EMN. These include the reservoir size N, the Input Scale IS, the Spectral Radius SR and the Sparsity α of the reservoir. We conducted experiments on four UCR datasets to illustrate the effects of these hyperparameters on EMN. We first consider the effect of the reservoir size N and evaluate the performance of EMN with different values of N. Then we evaluate the performance of EMN with different values of α while changing the IS and SR. The hyperparameters of EMN when applied to these four datasets are described below.

ECG200: The number of filters is set to 120 and the time scale is $0.6T$ & $0.7T$. For the experiments exploring the effects of reservoir size, the IS is set to 0.1, SR is 0.9 and α is 0.7.

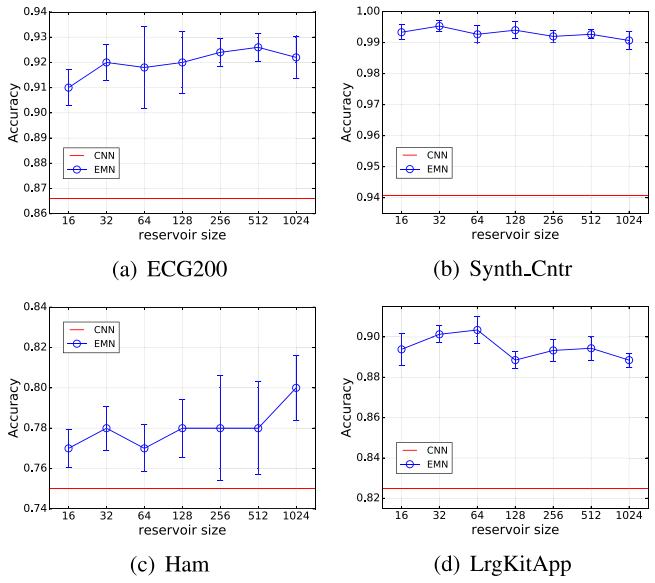
Synth_Cntr: The filter number is 150 and the time scale is $0.05T$ & $0.1T$. For the experiments exploring the effects of reservoir size, the IS is set to 0.1, SR is set to 0.9 and α is 0.7.

Ham: The filter number is 60 and the time scale is $0.025T$ & $0.05T$. For the experiments exploring the effects of reservoir size, the IS is set to 0.1, SR is set to 0.9 and α is 0.3.

LrgKitApp: The number of filters is set to 120 and the time scale is $0.05T$ & $0.1T$. For the experiments exploring the effects of reservoir size, the IS is set to 1, SR is set to 0.9 and α is 0.3.

The classification accuracies of EMN when increasing the value of N are shown in Fig. 13. For the ECG200 dataset and the Ham dataset, increasing the values of N gradually improves the accuracy. For the Synth_Cntr dataset, the accuracy is relatively stable when increasing the value of N. For the LrgKitApp dataset, the accuracy improves when the size of reservoir is less than 64, drops when N exceeds 64, and finally stabilizes. EMN always achieves a better result than the multi-scale CNN baseline. As shown in Fig. 13, setting reservoir size to 32 is enough to deal with the UCR univariate time series. Therefore, N is set to 32 in all the UCR time series datasets to make a trade-off between model accuracy and the number of model parameters.

The effects of other hyperparameters are shown in Fig. 14. The reservoir size is fixed at 32 for these experiments. In these figures, the value of sparsity α is varied from left to right, and within each panel, IS is chosen from $IS \in \{0.001, 0.01, 0.1, 0.5, 1\}$, while SR is chosen from $SR \in \{0.1, 0.5, 1, 1.5, 2\}$. As can be seen from Fig. 14, EMN always achieves quite good results when the value of IS is in the interval [0.1, 1]. And the accuracy decreases sharply when SR exceeds 1. Decreasing the value of α can slightly improve the accuracy and thus shows that a reservoir with sparse connections can enhance the capability of the reservoir to extract abundant temporal features.



(a) ECG200 (b) Synth_Cntr

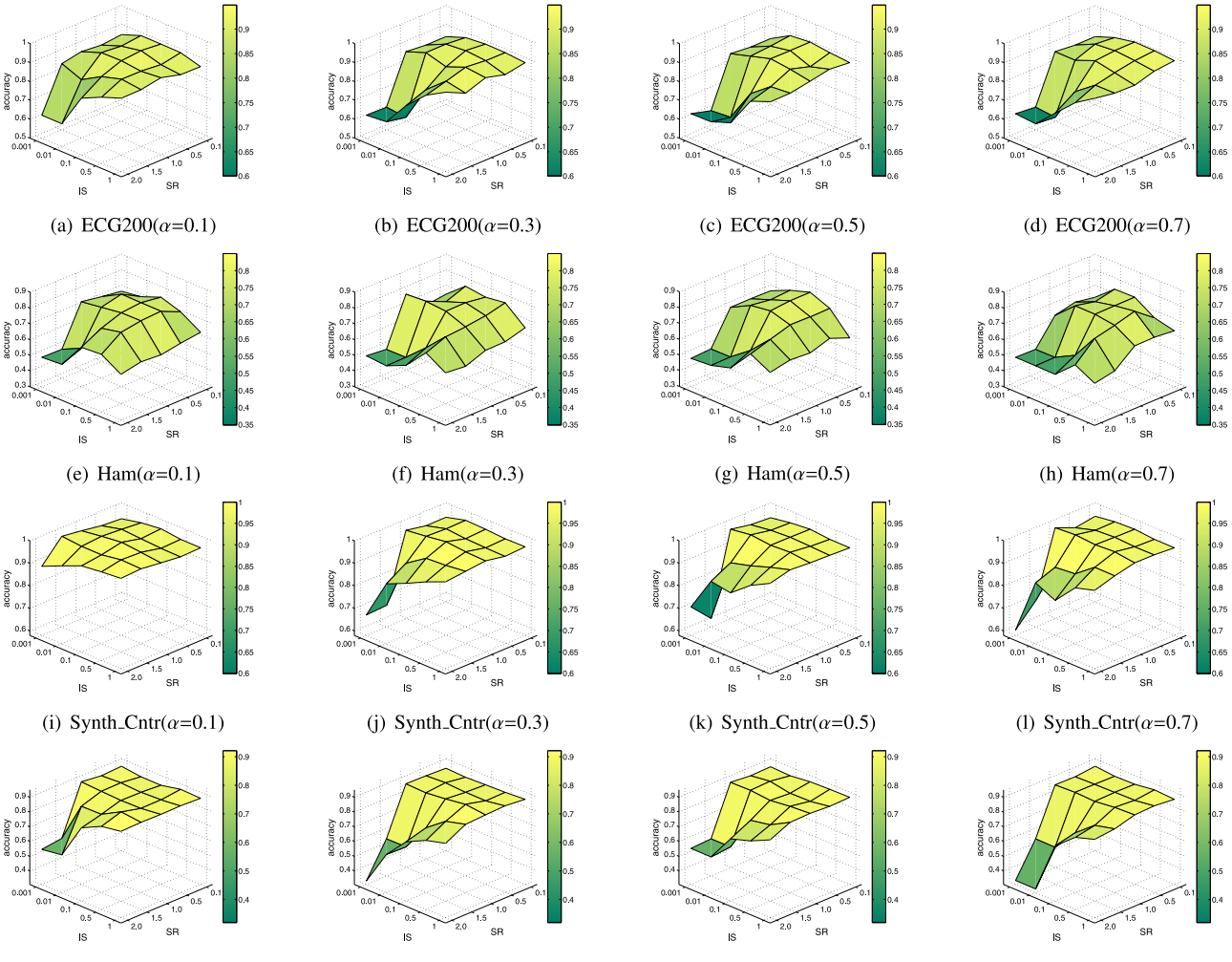
(c) Ham (d) LrgKitApp

Fig. 13. The classification accuracies of EMN and multi-scale CNN on 4 UCR datasets when increasing the size of reservoir.

6. Conclusion

In this paper, we propose a novel end-to-end model called EMN for time series classification tasks. The model can be extended to deal with 3D skeleton-based human action recognition by regarding skeleton joint sequences as multivariate time series. The core idea of EMN is to use a reservoir to encode the raw time series as an echo memory matrix, and then use multi-scale convolution and max-over-time pooling as a powerful decoder to extract the discriminative features.

The merits of using ESNs over other RNN/LSTM networks are twofold. First, the weights of the reservoir are randomly initialized and fixed; no training is required. The recurrent layer can be regarded as a high-dimensional (usually 100–1000D) nonlinear temporal kernel, and automatically provides abundant representations of the input time series. Second, due to the sparse connectivity of neurons in the reservoir, many loosely coupled oscillators are created, and information persists in one part of the reservoir without being propagated to other parts too quickly. This contributes to the short-term memory property of ESNs. Therefore, ESRs are very suitable and efficient in representing and capturing the temporal dynamics in time series. However, previous work that applied ESNs to TSC tasks did not achieve satisfactory results since they involved either training the model by predicting the next item of a sequence, or predicting the class label at each time step. The former is essentially a predictive



(a) ECG200(α=0.1) (b) ECG200(α=0.3) (c) ECG200(α=0.5) (d) ECG200(α=0.7)

(e) Ham(α=0.1) (f) Ham(α=0.3) (g) Ham(α=0.5) (h) Ham(α=0.7)

(i) Synth_Cntr(α=0.1) (j) Synth_Cntr(α=0.3) (k) Synth_Cntr(α=0.5) (l) Synth_Cntr(α=0.7)

(m) LrgKitApp(α=0.1) (n) LrgKitApp(α=0.3) (o) LrgKitApp(α=0.5) (p) LrgKitApp(α=0.7)

Fig. 14. The classification accuracies of the EMN with different α when changing the *IS* and *SR*. (The value of α increases from left to right.)

model, while the latter suffers from low classification accuracy since using the echo state at a single time step to determine the class label is difficult and inaccurate. In contrast to previous ESN-based classifiers, EMN collects the echo states at each time step into an echo memory matrix and uses multi-scale convolution to learn discriminative features from the echo memory. Hence, EMN determines the category label by considering the echo states at all time steps, which improves the classification accuracy.

By combining the merits of ESNs and CNNs, EMN is an efficient yet powerful framework, and provides an excellent choice for the combination of reservoir computing and the convolutional network paradigm. EMN achieves state-of-the-art results on both the UCR time series classification datasets and four 3D skeleton-based action recognition benchmarks, demonstrating the effectiveness of the proposed model.

Acknowledgments

The work described in this paper was partially funded by the National Natural Science Foundation of China (Grant Nos. 61502174, 61872148), the Natural Science Foundation of Guangdong Province (Grant No. 2017A030313355), the Guangzhou Science and Technology Planning Project (Grant Nos. 201704030051, 201902010020), the Opening Project of Guangdong Province Key Laboratory of Big Data Analysis and Processing (Grant No. 2017014). It was also supported by the National Science Foundation (USA) grant SMA 1041755 to the Temporal Dynamics of Learning Center, an NSF Science of Learning Center.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.neunet.2019.05.008.

References

Babinec, Š., & Pospíchal, J. (2006). Merging echo state and feedforward neural networks for time series forecasting. In *Artificial neural networks* (pp. 367–375).

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

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

Baydogan, M. G., & Runger, G. (2016). Time series representation and similarity based on local autopatterns. *Data Mining and Knowledge Discovery*, *30*(2), 476–509.

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

Berndt, D. J., & Clifford, J. (1994). Using dynamic time warping to find patterns in time series. In *Proceedings of the 3rd international conference on knowledge discovery and data mining* (pp. 359–370).

Boccato, L., Lopes, A., Attux, R., & Zuben, F. J. V. (2012). An extended echo state network using volterra filtering and principal component analysis. *Neural Networks*, *32*, 292–302.

Boccato, L., Soriano, D. C., Attux, R., & Zuben, F. J. V. (2012). Performance analysis of nonlinear echo state network readouts in signal processing tasks. In *The 2012 international joint conference on neural networks* (pp. 1–8).

Butcher, J., Verstraeten, D., Schrauwen, B., Day, C., & Haycock, P. (2010). Extending reservoir computing with random static projections: a hybrid between extreme learning and RC. In *European symposium on artificial neural networks, 18th, proceedings* (pp. 303–308).

Chatzis, S. P., & Demiris, Y. (2011). Echo state gaussian process. *IEEE Transactions on Neural Networks*, *22*(9), 1435–1445.

Chen, Y., Keogh, E., Hu, B., Begum, N., Bagnall, A., Mueen, A., & Batista, G. (2015). The UCR time series classification archive. www.cs.ucr.edu/~eamonn/time_series_data/.

Chen, C., Liu, M., Zhang, B., Jiang, J., Jiang, J., & Liu, H. (2016). 3D action recognition using multi-temporal depth motion maps and fisher vector. In *Proceedings of the twenty-fifth international joint conference on artificial intelligence* (pp. 3331–3337).

Chen, H., Tang, F., Tino, P., Cohn, A. G., & Yao, X. (2015). Model metric co-learning for time series classification. In *Proceedings of the 24th international conference on artificial intelligence* (pp. 3387–3394).

Cho, K., & Chen, X. (2014). Classifying and visualizing motion capture sequences using deep neural networks. In *2014 International conference on computer vision theory and applications: Vol. 2* (pp. 122–130).

Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of Machine Learning Research (JMLR)*, *12*(Aug), 2493–2537.

Cui, Z., Chen, W., & Chen, Y. (2016). Multi-scale convolutional neural networks for time series classification. arXiv preprint arXiv:1603.06995.

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

Deng, H., Runger, G., Tuv, E., & Vladimir, M. (2013). A time series forest for classification and feature extraction. *Information Sciences*, *239*, 142–153.

Devanne, M., Wannous, H., Berretti, S., Pala, P., Daoudi, M., & Del Bimbo, A. (2013). Space-time pose representation for 3D human action recognition. In *New trends in image analysis and processing* (pp. 456–464).

Du, Y., Wang, W., & Wang, L. (2015). Hierarchical recurrent neural network for skeleton based action recognition. In *2015 IEEE conference on computer vision and pattern recognition* (pp. 1110–1118).

Gallicchio, C., & Micheli, A. (2011). Architectural and markovian factors of echo state networks. *Neural Networks*, *24*(5), 440–456.

Goodfellow, I., Bengio, Y., Courville, A., & Bengio, Y. (2016). *Deep learning*. The MIT Press.

Górecki, T., & Łuczak, M. (2013). Using derivatives in time series classification. *Data Mining and Knowledge Discovery*, *26*(2), 310–331.

Górecki, T., & Łuczak, M. (2014). Non-isometric transforms in time series classification using dtw. *Knowledge-Based Systems*, *61*, 98–108.

Gowayyed, M. A., Torki, M., Hussein, M. E., & El-Saban, M. (2013). Histogram of oriented displacements (HOD): describing trajectories of human joints for action recognition. In *Proceedings of the twenty-third international joint conference on artificial intelligence* (pp. 1351–1357).

Grabocka, J., Schilling, N., Wistuba, M., & Schmidt-Thieme, L. (2014). Learning time-series shapelets. In *Proceedings of the 20th ACM SIGKDD international conference on knowledge discovery and data mining* (pp. 392–401).

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

Bin Huang, G., Yu Zhu, Q., & Kheong Siew, C. (2004). Extreme learning machine: A new learning scheme of feedforward neural networks. In *Proceedings of international joint conference on neural networks: Vol. 2* (pp. 985–990).

Hussein, M. E., Torki, M., Gowayyed, M. A., & El-Saban, M. (2013). Human action recognition using a temporal hierarchy of covariance descriptors on 3D joint locations. In *Proceedings of the twenty-third international joint conference on artificial intelligence: vol. 13* (pp. 2466–2472).

Jaeger, H. (2001). The "echo state" approach to analysing and training recurrent neural networks-with an erratum note. *GMD Technical Report 148*, (p. 13). Bonn, Germany: German National Research Center for Information Technology.

Jaeger, H., & Haas, H. (2004). Harnessing nonlinearity: predicting chaotic systems and saving energy in wireless communication. *Science*, *304*(5667), 78–80.

Kerola, T., Inoue, N., & Shinoda, K. (2014). Spectral graph skeletons for 3D action recognition. In *Asian conference on computer vision* (pp. 417–432).

Kingma, D. P., & Ba, J. (2014). Adam: a method for stochastic optimization. arXiv preprint arXiv:1412.6980.

Lecun, Y., Bottou, L., Bengio, Y., & Haffner, P. (1998). Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, *86*(11), 2278–2324.

Li, D., Han, M., & Wang, J. (2012). Chaotic time series prediction based on a novel robust echo state network. *IEEE Transactions on Neural Networks and Learning Systems*, *23*(5), 787–799.

Li, W., Zhang, Z., & Liu, Z. (2010). Action recognition based on a bag of 3d points. In *2010 IEEE computer society conference on computer vision and pattern recognition - workshops* (pp. 9–14).

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

Lo Presti, L., La Cascia, M., Sclaroff, S., & Camps, O. (2015). Hankelet-based dynamical systems modeling for 3D action recognition. *Image and Vision Computing*, *44*(C), 29–43.

Lukoševičius, M. (2007). Echo state networks with trained feedbacks. *Technical report 4, Technical report*.

Lukoševičius, M. (2012). A practical guide to applying echo state networks. In *Neural networks: Tricks of the trade* (pp. 659–686). Springer.

Lukoševičius, M., & Jaeger, H. (2009). Reservoir computing approaches to recurrent neural network training. *Computer Science Review*, *3*(3), 127–149.

Müller, M., Röder, T., Clausen, M., Eberhardt, B., Krüger, B., & Weber, A. (2007). Documentation mocap database hdm05.

Qiao, J., Li, F., Han, H., & Li, W. (2017). Growing echo-state network with multiple subreservoirs. *IEEE Transactions on Neural Networks and Learning Systems*, *28*(2), 391–404.

Rosenblatt, F. (1958). The perceptron: a probabilistic model for information storage and organization in the brain. *Psychological Review*, *65*(6), 386–408.

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

Seidenari, L., Varano, V., Berretti, S., Bimbo, A. D., & Pala, P. (2013). Recognizing actions from depth cameras as weakly aligned multi-part bag-of-poses. In *2013 IEEE conference on computer vision and pattern recognition workshops* (pp. 479–485).

Shi, Z., & Han, M. (2007). Support vector echo-state machine for chaotic time-series prediction. *IEEE Transactions on Neural Networks*, *18*(2), 359–372.

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. arXiv preprint arXiv:1409.1556.

Skowronski, M. D., & Harris, J. G. (2007). Automatic speech recognition using a predictive echo state network classifier. *Neural Networks*, *20*(3), 414–423.

Slama, R., Wannous, H., Daoudi, M., & Srivastava, A. (2015). Accurate 3D action recognition using learning on the grassmann manifold. *Pattern Recognition*, *48*(2), 556–567.

Steinier, J., Termonia, Y., & Deltour, J. (1972). Smoothing and differentiation of data by simplified least square procedure. *Analytical Chemistry*, *44*(11), 1906–1909.

Sukhbaatar, S., Weston, J., Fergus, R., et al. (2015). End-to-end memory networks. In *Advances in neural information processing systems* (pp. 2440–2448).

Triefenbach, F., Jalalvand, A., Demuynck, K., & Martens, J. P. (2013). Acoustic modeling with hierarchical reservoirs. *IEEE Transactions on Audio, Speech, and Language Processing*, *21*(11), 2439–2450.

Vemulapalli, R., Arrate, F., & Chellappa, R. (2014). Human action recognition by representing 3D skeletons as points in a lie group. In *2014 IEEE conference on computer vision and pattern recognition* (pp. 588–595).

Verstraeten, D., Schrauwen, B., dHaene, M., & Stroobandt, D. (2007). An experimental unification of reservoir computing methods. *Neural Networks*, *20*(3), 391–403.

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

Williams, R., & Zipser, D. (1989). A learning algorithm for continually running fully recurrent neural networks. *Neural Computation*, *1*, 270–280.

Xia, L., Chen, C. C., & Aggarwal, J. K. (2012). View invariant human action recognition using histograms of 3d joints. In *2012 IEEE computer society conference on computer vision and pattern recognition workshops* (pp. 20–27).

Xia, Y., Jelfs, B., Hulle, M. M. V., Principe, J. C., & Mandic, D. P. (2011). An augmented echo state network for nonlinear adaptive filtering of complex noncircular signals. *IEEE Transactions on Neural Networks*, *22*(1), 74–83.

Ye, L., & Keogh, E. (2009). Time series shapelets: a new primitive for data mining. In *Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 947–956). ACM.

Yildiz, I. B., Jaeger, H., & Kiebel, S. J. (2012). Re-visiting the echo state property. *Neural Networks*, *35*, 1–9.

Zhang, X., Wang, Y., Gou, M., Sznaier, M., & Camps, O. (2016). Efficient temporal sequence comparison and classification using gram matrix embeddings on a riemannian manifold. In *2016 IEEE conference on computer vision and pattern recognition* (pp. 4498–4507).

Zhu, Y., Chen, W., & Guo, G. (2013). Fusing spatiotemporal features and joints for 3D action recognition. In *2013 IEEE conference on computer vision and pattern recognition workshops* (pp. 486–491).

Zhu, W., Lan, C., Xing, J., Zeng, W., Li, Y., Shen, L., & Xie, X. (2016). Co-occurrence feature learning for skeleton based action recognition using regularized deep LSTM networks. In *Proceedings of the thirtieth AAAI conference on artificial intelligence* (pp. 3697–3703).