






Engineering Applications of Artificial Intelligence

Volume 122, June 2023, 106151

Self-Attention Causal Dilated Convolutional Neural Network for Multivariate Time Series Classification and Its Application

Wenbiao Yang , Kewen Xia , Zhaocheng Wang , Shurui Fan , Ling Li 

Show more 

 Outline |  Share  Cite

<https://doi.org/10.1016/j.engappai.2023.106151>

[Get rights and content](#)

Abstract

Time Series Classification (TSC) in data mining is gradually developing as an important research direction. Many researchers have developed an extensive interest in Multivariate Time Series Classification (MTSC). The Self-Attention Causal Dilated Convolutional Neural Network (SACDCNN) is proposed to address the limitations of existing models that perform poorly on classification tasks. It designs the residual and dense blocks based on Causal Dilated Convolution based on the traditional residual and dense networks that still have superior performance after deepening the network hierarchy and the dependence of time series on long-range information. A Self-Attention mechanism (SA) is also incorporated to extract the internal autocorrelation of time series features. Comparison experiments on 20 benchmark University of California, Riverside (UCR) and University of California, Irvine (UCI) datasets with eight high-performance classification models show that the method can improve the classification accuracy of time series datasets. Finally, it was applied to petroleum logging reservoir recognition, and a comparison experiment was conducted on two wells. The results show that SACDCNN is effective and significantly superior. It overcomes the shortcomings of traditional logging interpretation techniques and improves the efficiency and success rate of oil and gas exploration.

 Previous

Next 

Keywords

Multivariate Time Series Classification; Causal Dilated Convolution; Convolutional Neural Network; Self-Attention Mechanism; Petroleum logging reservoir recognition

1. Introduction

Time series data are involved in various research areas, from ECG signals ([Maharaj and Alonso, 2014](#)) to weather readings ([Ayhan and Samet, 2016](#)), and time-series information is distributed in all aspects of people's daily lives. A temporal signal is usually a series of data points sampled at uniform intervals. With the rapid innovation of sensor technology, the amount of time series data it collects is increasing exponentially. Time series data sets are divided into univariate and Multivariate Time Series (MTS). A univariate time series is defined as a set of measurement series from the same variable. Similarly, An MTS is defined as a sequence of measurements collected from multiple variables or sensors.

In the last decade, MTSC tasks have received much attention. It is mainly used in the fields of behavior recognition ([Alawneh et al., 2021](#)), medically assisted diagnosis ([Bhaskar et al., 2021](#)), and sentiment classification ([Naqvi et al., 2020](#)). Many researchers have come up with valuable ideas for TSC. Distance-based methods and K-Nearest Neighbors (KNN) are widely accepted in MTSC ([Orsenigo and Vercellis, 2010](#)). Many studies have shown that Dynamic Time Warping (DTW) is the best distance-based metric to use with KNN ([Seto et al., 2015](#)). There is a different way of thinking besides distance-based metrics. It is generally believed that feature-based classification algorithms are difficult to capture the inherent features of time series data ([Zheng et al., 2014](#)). However, two recognized feature-based algorithms, namely Hidden Conditional Random Fields (HCRF) and Hidden Unit Logic Model (HULM), have been successfully applied to activity recognition scenarios ([Pei et al., 2018](#)).

For MTSC, researchers have proposed the application of feature dimensionality reduction or joining MTS dimensions into a univariate time series. Baydogan et al. proposed Learning Pattern Similarity (LPS), which is a similar model for extracting segments from MTS ([Baydogan and Runger, 2016](#)). Wistuba et al. proposed Ultra-Fast Shapelets (UFS), which derives random Shapelets from MTS and employs a linear Support Vector Machine (SVM) or Random Forest (RF) classifier ([Wistuba et al., 2015](#)). Cuturi et al. proposed an AR kernel-based distance metric for the classification of MTS ([Cuturi and Doucet, 2011](#)). Tuncel et al. proposed an autoregressive forest, which uses a collection of trees trained with different time lags ([Tuncel and Baydogan, 2018](#)). Singh et al. proposed the first work of its kind, which combines primitives of parallel implementations of k-nearest neighbors (k-NN) with population based metaheuristic algorithms to accelerate the calibration process of a classification model named eNN10 ([Singh et al., 2017](#)). To improve the quality of the resulting clusters, Borlea et al. proposed a way of improving the resulted clusters generated by the K-means algorithm by post processing the resulted clusters with a supervised learning algorithm (

[Borlea et al., 2022](#)). To address the many difficulties associated with object detection, Arican & Aydin proposed a system that combines Bag of Visual Words (BoVW) with the SURF and depth information to create a new 3D descriptor ([Arican and Aydin, 2022](#)).

With the rapid development of deep learning, it has also achieved excellent results in MTSC. Karim et al. proposed Long Short Term Memory Fully Convolutional Network (LSTM-FCN) and Attention LSTM-FCN (ALSTM-FCN), and could further improve the accuracy by using squeezed and excited blocks ([Karim et al., 2019](#)). Yang et al. proposed Attentional Gated ResNet, which uses layered residual connections to achieve multiscale perceptual fields and capture multi-granularity temporal information ([Yang et al., 2022](#)). Zheng et al. proposed a Multichannel Deep Convolutional Neural Network (MDCNN), which learns features from a single univariate time series of each channel and then fuses the information from all channels for feature fusion ([Zheng et al., 2016](#)). Bai et al. proposed a Temporal Convolutional Network (TCN) in 2018, which includes causal convolution, dilated convolution, and residual chunking and can effectively solve the extraction of MTS information ([Bai et al., 2018](#)). Three simple but powerful baselines are proposed by Wang et al. The first is a Fully Connected Convolutional Neural Network (FCN) consisting of three convolutional blocks, then a primary Multi-layer Perceptron (MLP) implemented by stacking three fully connected layers. Finally, a relatively deep Residual Network (Resnet) is also proposed for MTSC tasks ([Wang et al., 2016](#)). Tanisaro et al. proposed the Time Warped Invariant Echo State Network (Twiesn), which can select the appropriate network size based on the amount of training data and adaptively choose the spectral radius and input weight ratio ([Tanisaro and Heidemann, 2017](#)). Serrà et al. proposed two variants of the Encoder. The first approach is to train the model in an end-to-end manner using the target dataset. While the second approach performs pre-training on the source dataset, the role of the target dataset is to fine-tune the network parameters ([Serrà et al., 2018](#)). Zhao et al. proposed a new CNN-based TSC method, which can automatically mine and extracts the internal correlation of the input time series and generate depth features for classification ([Zhao et al., 2017](#)). Ismail Fawaz et al. proposed Inception based on the recent success of Inception-based networks used for various computer vision tasks, which is also integrated by a set of deep CNN models ([Ismail Fawaz et al., 2020](#)). Due to the significant advantages of transformers and CNNs in feature extraction, Wu et al. proposed the Multivariate time series classification Convolutional Transformer Network (MCTNet), which can capture the potential depth information in multivariate time series more accurately ([Wu et al., 2022](#)). To utilize labels effectively, Hao et al. proposed **Mixed Supervised Contrastive Loss** (MSCL), which is computed by mixing self-supervised, intra-class, and inter-class supervised comparative learning methods ([Hao et al., 2023](#)). Lyu et al. proposed the Multiscale Echo Self-Attention Memory Network (MESAMN) to address the problem that Echo State Network cannot capture long-term dependency information well ([Lyu et al., 2023](#)). [Table 1](#) is a literature review of research in the field of MTSC, and the bolded parts of the table are the innovations of this paper.

With the development and needs of complex oil and gas exploration and geological engineering evaluation, it is necessary to adopt artificial intelligence technology to provide new concepts for engineering practice ([Zhai et al., 2016](#)). In particular, the logging reservoir identification task is an

important factor in oil and gas logging evaluation. It is an advanced stage of reducing logging information into geological information and reflecting logging interpretation results and application values, which will directly affect the efficiency and success rate of oil and gas exploration. However, for complex lithologic formations, fractured reservoirs, low porosity, low resistivity and other types of reservoirs, the conventional logging reservoir interpretation method is time-consuming, requires high competence of engineers and has too many human factors interfering. For this reason, this paper applies the proposed SACDCNN to logging reservoir identification, which is very meaningful for decision making in logging interpretation, oil and gas exploration and development, and geoengineering evaluation.

Table 1. Literature review of research in the field of MTSC.

Author(s) and Year	Method	Key characteristics
Karim et al. (2019)	LSTM-FCN and ALSTM-FCN	Use squeezed and excited blocks
Yang et al. (2022)	Attentional Gated ResNet	Capture multi granularity temporal information
Zheng et al. (2016)	MDCNN	Independent convolutional pooling operation for each dimension of multidimensional time-series data
Bai et al. (2018)	TCN	Causal convolution, dilated convolution, and residual chunking
Wang et al. (2016)	MLP	Simple and easy to implement, but the generalization of the model is poor
Wang et al. (2016)	FCN	The parameters of the model are set independent of the length of the input data and are suitable for migration learning
Wang et al. (2016)	Resnet	The direct connection of residuals greatly reduces the problem of gradient disappearance in model training
Tanisaro and Heidemann (2017)	Twiesn	Adaptive selection of spectral radius and input weighting ratio
Serrà et al. (2018)	Encoder	Model parameter setting does not depend on data dimensionality
Zhao et al. (2017)	CNN	The number of parameters in the model is reduced
Ismail Fawaz et al. (2020)	Inception	Increase feature expressiveness while reducing computation
Wu et al. (2022)	MCTNet	Different advantages of convolutional neural networks and self-attentiveness are exploited
Hao et al. (2023)	MSCL	Capturing multi-scale contextual information, hierarchically computed at the timestamp level
Lyu et al. (2023)	MESAMN	Capture time series features long-term dependencies

Our proposal	SACDCNN	Deep network depth, efficient capture of long-distance temporal features, and low memory consumption
---------------------	----------------	---

By reviewing the key features of previous works, it is found that they differ in their treatment of time series feature extraction. In particular, the Causal convolution and Dilated Convolution proposed by TCN provides new ideas for the MTSC task. The Self-Attention mechanism shows excellent performance in feature extraction for both image and temporal sequences. Our motivation is the design of residuals and dense blocks based on the Causal Dilated Convolution, and incorporates Self-Attention mechanism, which can effectively extract the internal autocorrelation of time series features.

For the limitations of the existing models that do not perform well on the classification task, the SACDCNN is proposed. It is inspired by the characteristics of traditional residual and dense networks that maintain superior performance after network hierarchy deepening and the characteristics of long-distance information dependence of time series and designs residual and dense blocks based on Causal Dilated Convolution. It also incorporates a Self-Attention mechanism to extract the internal autocorrelations of time series features. Twenty benchmark datasets were obtained from UCR/UCI, and independent comparison experiments were conducted with TCN, FCN, MLP, Resnet, Twiesn, Encoder, CNN, and Inception. And the results were tested nonparametrically, and the results of each experiment showed that SACDCNN showed significant superiority and stability in the MTSC task. We also meet the current urgent need for deep learning in logging interpretation by applying SACDCNN to logging reservoir recognition. Through comparison experiments on real oil and gas well datasets, the results show that SACDCNN outperforms other listed comparison models in several evaluation metrics, and the oil and gas formation recognition results remain highly consistent with the real oil and gas formation distribution.

The main contributions of our work are summarized as follows.

- (1) The current state of research on MTSC tasks is reviewed, and the strengths and weaknesses of the publicly available DNN models are analyzed. The motivation for proposing SACDCNN is described.
- (2) The innovation of this paper is the proposed SACDCNN. SACDCNN is proposed to address the problem that existing DNN models perform poorly on time series classification tasks. SACDCNN not only design residuals and dense blocks based on Causal Dilated Convolution, but also incorporate Self-Attention mechanism. They are used to extract important features of the time series, which is beneficial to improve the performance of the classifier.
- (3) In the experimental part, nine high-performance DNN models were evaluated on 20 benchmark datasets. Various validity methods were used to evaluate the effectiveness and superiority of SACDCNN.
- (4) SACDCNN is applied to the logging reservoir recognition task, and practical applications are

carried out in two wells. SACDCNNN has significant superiority over other comparative models and can identify the distribution of logging oil and gas reservoirs more accurately. SACDCNNN effectively addresses practical industrial needs and is of great practical importance.

The rest of the paper is organized as follows. Section 2 presents the background work on causal expansion convolution, Resnet basic unit, and Densenet basic unit. A detailed description of the proposed SACDCNN is given in Section 3. The comparative experiments and results analysis of each model are presented in Section 4. The setup of the experiments for petroleum logging engineering applications is presented in Section 5, and the experimental results are discussed. Finally, Section 6 concludes the paper and presents information for future work.

2. Background works

2.1. Causal Dilated Convolution

Before TCN, RNNs such as LSTM and GRU were often associated to new sequence modeling tasks. However, Bai et al. demonstrated that TCN can efficiently handle the sequence modeling task even better than other models (Bai et al., 2018). Bai et al. also verified that TCN maintains more extended memory than LSTM. Causal Dilated Convolution in TCN has the advantages of parallel processing, flexible sense field size, and variable length input.

2.1.1. Causal convolution

The core of the convolutional neural network is the feature extraction role of the convolutional layer. For the characteristics of time series, Causal Convolution considers that the output information depends only on the past input information, thus effectively avoiding the interference of future information, and thus, it is a one-way architecture. The architecture of causal convolution is shown in Fig. 1. To better explain Fig. 1, our mathematical modeling of it is as follows.

If the input time series are $X = (x_1, x_2, \dots, x_T)$, $X \in \mathbb{R}^n$ and the filter is $F = (f_1, f_2, \dots, f_K)$, the output of x_T after causal convolution is the following equation.

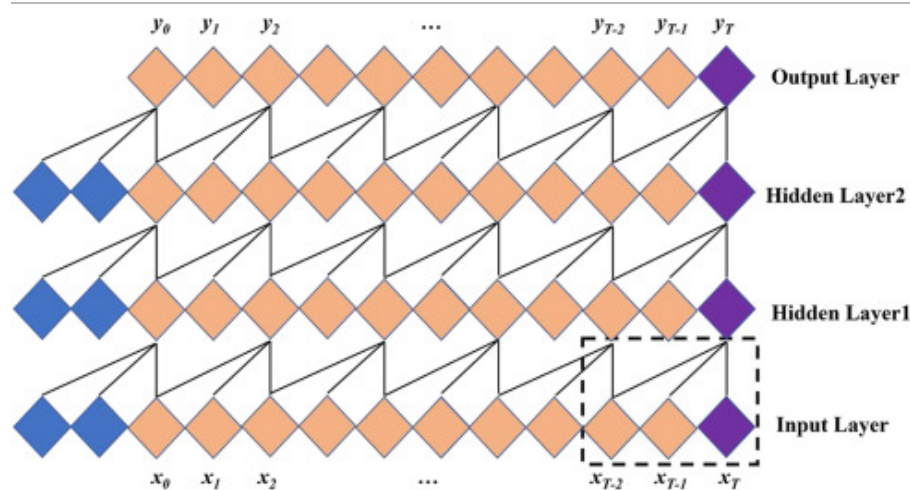
$$Y(T) = (X * f)(T) = \sum_{i=0}^{k-1} F(i) \cdot x_{T-i} \quad (1)$$

where k is the filter size, and $T - i$ accounts for the direction of the past.

For Causal Convolution, the value of the moment t in its previous layer depends only on the moment t and the previous value in the next layer. Unlike traditional Convolutional Neural Networks, causal convolution cannot see future data, and it is a one-way structure rather than a two-way one. In other words, only the previous cause can have the later result, which is a strictly time-constrained model, hence it is called causal convolution.

However, Causal Convolution is limited by the size of the perceptual field and performs well in predicting information with short memory histories. For dealing with very long time series problems,

it is necessary to increase the depth of the network to ensure better output information.



[Download: Download high-res image \(388KB\)](#)

[Download: Download full-size image](#)

Fig. 1. Architecture of causal convolution.

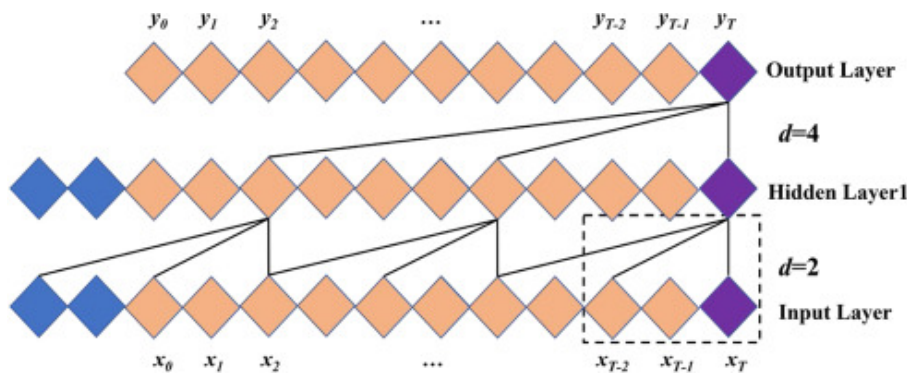
2.1.2. Dilated convolution

Dilated Convolution is proposed to solve the problem of a restricted field of perception for causal convolution. Unlike the causal convolution, the Dilated Convolution allows the input to be sampled at intervals, i.e., the dilated factor d . The dilated factor at the lowest level is 1, i.e., each piece of information is sampled for the input, i.e., the same as the Causal Convolution. When $d = 2$, it also means that sampling every 2 points is performed at this point. The SACDCNN can effectively expand the perceptual field so that the output of the top layer can receive a larger range of input information. In addition, by processing the same filters in parallel in each layer, the computational efficiency of the whole model can be improved. The Dilated Convolution architecture is shown in Fig. 2. To better explain Fig. 2, our mathematical modeling of it is as follows.

If the input time series is $X = (x_1, x_2, \dots, x_T)$, $X \in \mathbb{R}^n$ and the filter is $F = (f_0, f_1, \dots, f_{k-1})$, the output of x_T after Dilated Convolution with a dilation factor is the following equation.

$$Y(T) = (X *_d f)(T) = \sum_{i=0}^{k-1} F(i) \cdot x_{T-d \cdot i} \quad (2)$$

where d is the dilated factor, k is the filter size, and $T - d \cdot i$ accounts for the direction of the past.



[Download: Download high-res image \(278KB\)](#)

[Download: Download full-size image](#)

Fig. 2. Architecture of dilated convolution.

2.2. Resnet basic unit

Highway Network proposed by Srivastava et al. introduces Gated Shortcut Connections. These parameterized gates control how much information is allowed to flow through the shortcut channels. Highway Network was inspired by the “LSTM structure proposed to solve the problem of RNN”, that is, the addition of “gate” structure, Highway Network mainly solves the problem of deepening the network, the gradient information flow back is blocked, resulting in network training difficulties (Srivastava et al., 2015). Resnet has some similarities with the Highway Network idea but improves it by proposing residual blocks and retaining Shortcut connections. The problem of gradient disappearance due to increasing number of layers in neural networks is alleviated (He et al., 2016).

If the target mapping of the residual unit in the original Resnet is $H(x)$, then the target mapping will be difficult to learn. Resnet makes it impossible for the residual unit to learn the target mapping directly, but by learning the residual $F(x) = H(x) - x$. Resnet consists of residual units. The construction of each residual unit consists of a convolutional layer, a batch normalization layer, and a nonlinear activation function layer. Assuming that the input of the n th residual cell is X , the input of the next layer is given by the following equation.

$$H(X) = \text{Activation}[X + F_l(X)] \quad (3)$$

where $F_l(\bullet)$ is the nonlinear transformation function, $\text{Activation}[\bullet]$ represents the nonlinear activation function.

2.3. Densenet basic unit

The Densenet network was proposed in 2017, borrowing the advantages of Resnet network and GoogleNet, and from the perspective of optimal features, the Densenet model proposes a new dense connection mechanism compared to the classical Resnet network (Huang et al., 2017). Each layer of the Resnet network is short-circuited with the preceding 2 to 3 layers, and the connections are

summed at the element level. In Densenet, however, each layer is connected to all previous layers in the channel dimension and serves as input to the next layer.

For an L -layer network, the Densenet model contains a total of $\frac{L(L+1)}{2}$ connections, which is more densely connected than the Resnet network. The Densenet network directly connects the feature maps from different layers, which can effectively achieve feature reuse and improve computational efficiency.

In a conventional network, the output of layer l is the following equation.

$$x^l = F_l(x^{l-1}) \quad (4)$$

where $F_l(\cdot)$ is the nonlinear conversion function.

For the Resnet network, the output of layer l is the following equation due to the addition of the input from the previous network layers.

$$x^l = F_l(x^{l-1}) + x^{l-1} \quad (5)$$

In Densenet, the features extracted from all previous layers are added as input, and then the output is the following equation.

$$x^l = F_l([x^0, x^1, \dots, x^{l-1}]) \quad (6)$$

3. Self-attention causal dilated convolutional neural network

3.1. Self-attention mechanism

The attention mechanism (Vaswani et al., 2017) was originally used in the field of computer vision to propose useful features in sparse data. In 2017, the attention mechanism was first applied to the field of natural language processing by the Google Machine Translation team, which employed the Self-Attention mechanism to perform a text representation learning task and achieved excellent performance. The Self-Attention mechanism module is shown in Fig. 3. This module calculates different weights between each element of the sequence (both long-term and short-term) and captures dynamic long-term and short-term preferences. It has been argued in numerous literatures that the module can adaptively learn the dependencies of short- and long-term sequential behavior.

The input to the Self-Attention Mechanism module consists of three parts: Query, Key And Value. The output is a weighted sum based on the similarity of Query and Key. Query with dimension d_k is obtained by a linear transformation of embedding $X = [x_1, \dots, x_n]$ on a set of inputs with dimension d_{model} :

$$Q = W_q \cdot X \quad (7)$$

In addition, the Key and Value with dimension d_v can be calculated as follows.

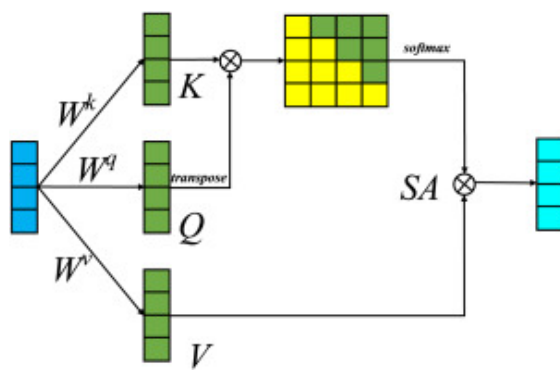
$$K = W_k \cdot X \quad (8)$$

$$V = W_v \cdot X \quad (9)$$

The Self-Attention mechanism calculates the similarity between the Query and all Key, and then divides by the scaling factor $\sqrt{d_k}$. The Softmax function is then used to obtain the weights of each value. We use the cosine function to calculate the similarity. The output of SA can be calculated as follows.

$$\text{Similarity}(Q, K) = \frac{Q \cdot K}{\|Q\| \cdot \|K\|} \quad (10)$$

$$SA_t(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \quad (11)$$



Download: [Download high-res image \(85KB\)](#)

Download: [Download full-size image](#)

Fig. 3. Self-attention mechanism module.

3.2. Residual block structure design based on Causal Dilated Convolution

In general, as the depth of the network increases, it is more beneficial for feature extraction, which means that the learning effect will be better. However, in practical experimental validation, it is found that the continuous deepening of the network layers may instead lead to a decrease in network performance. The proposers of Resnet point out that the blind superposition of convolutional and pooling layers in the network often tends to lead to the problem of gradient disappearance or gradient explosion, as well as the degradation problem.

The factors that affect the perceptual field of SACDCNN are network depth n , filter size k and dilated factor d . Therefore, it is necessary to construct a deep SACDCNN. SACDCNN uses a generic residual module, the design of which is shown in Fig. 4. Suppose the input of the i th residual block is X , then the output after causal expansion convolution is $F(X)$. Then the output of this residual block is $H(X)$. As can be seen from Fig. 4, Residual Block consists of two sets of Causal Dilated Convolutional layers, weight normalization layers, activation function LeakyRelu layers, and Dropout layers connected sequentially from the bottom to the top. The input of the first Causal Dilated Convolution layer is used as the input of the second Causal Dilated Convolution after passing through the normalization

layer, the activation function LeakyRelu layer and the Dropout layer in turn. The feature data are sequentially passed through the normalization layer, the activation function LeakyRelu layer and the Dropout layer of the second causal dilated convolution layer to obtain the output.

A series of temporal convolution blocks are defined, each of which contains a sequence of causal dilated convolution layers. The activations between the l th layer and j th block are given by $S^{(j,i)} \in \mathfrak{R}^{F_w \times T}$. Note that each layer has the same number of filters F_w . This enables us to combine activations from different layers later. Each layer consists of a set of causal dilated convolutions with dilation parameter d , a non-linear activation $f(\cdot)$, and a residual connection which combines the layer's input and the causal dilated convolution output. Convolutions are only applied over two-time steps, t and $t-d$. The filters are parameterized by $W = \{W^{(1)}, W^{(2)}\}$ with $W^{(i)} \in \mathfrak{R}^{F_w \times F_w}$ and bias vector $b \in \mathfrak{R}^{F_w}$. Suppose $\hat{S}_t^{(j,l)}$ be the output of the causal dilated convolution at moment t and $\hat{S}_t^{(j,l)}$ be the output after adding the residual connection. The specific complete equation can be written as follows.

$$\hat{S}_t^{(j,l)} = f(W^{(1)} S_{t-s}^{(j,l-1)} + W^{(2)} S_t^{(j,l-1)} + b) \quad (12)$$

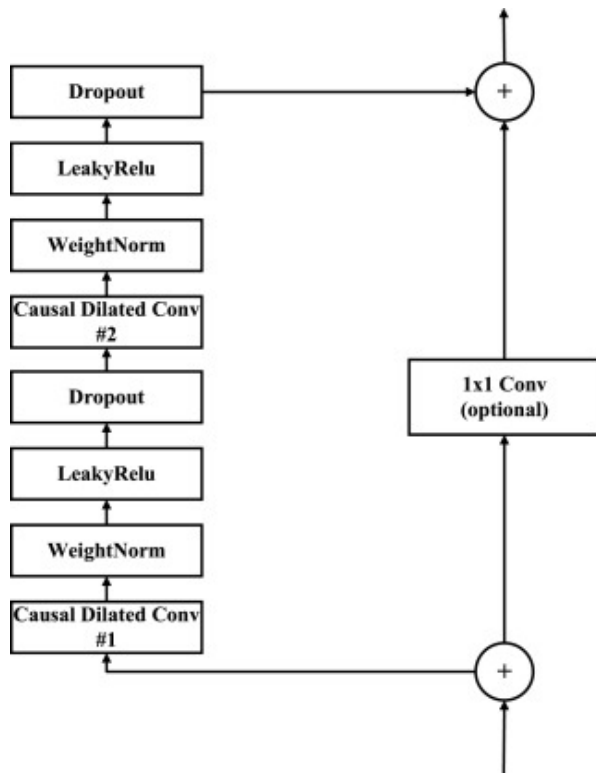
$$S_t^{(j,l)} = S_t^{(j,l-1)} + V \hat{S}_t^{(j,l)} + e \quad (13)$$

where $V \in \mathfrak{R}^{F_w \times F_w}$ and $e \in \mathfrak{R}^{F_w}$ represent a set of weights and biases for the residual respectively. Note that W, b, V, e are separate for each layer.

The output of each block $Z_t \in \mathfrak{R}^{F_w \times F_w}$ is summed using a set of skip connections as follows:

$$RZ_t = \text{LeakyRelu}\left(\sum_{j=1}^B S_t^{(j,L)}\right) \quad (14)$$

where B is the number of Causal Dilated Convolution.



Download: [Download high-res image \(140KB\)](#)

Download: [Download full-size image](#)

Fig. 4. Basic residual block architecture.

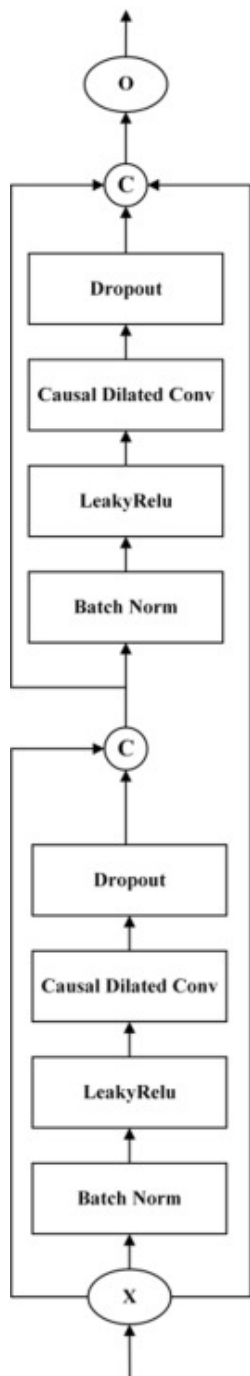
3.3. Design of dense block structure based on Causal Dilated Convolution

In Dense Block, the feature maps of each layer are of the same size and can be connected in the channel dimension. Assuming that the number of channels of the feature map of the input layer is k_0 , and each layer in Dense Block outputs k feature maps after convolution, i.e., the number of channels of the feature map obtained is k , then the number of channels of the input layer L is $k_0 + (L - 1)k$. We call k the growth rate of the network.

The basic Dense Block architecture is shown in Fig. 5. As can be seen from Fig. 5, it consists of two basic units, each of which consists of BatchNorm, LeakyRelu, Causal Dilated Convolution and Dropout layers in turn. Each layer accepts the feature maps of all the previous layers, i.e., the features are passed to the next layer after directly contacting the features of all the previous layers.

The convolution operation in this section is the same as in the previous section, so Eqs. (12), (13) still represent the convolution process in this section. The only difference is that the final output is:

$$DZ_t = \text{LeakyReLU}\left(\left[S_t^{(1,L)}, \dots, S_t^{(j,L)}\right]\right), j = 2 \cdots B \quad (15)$$



[Download: Download high-res image \(120KB\)](#)

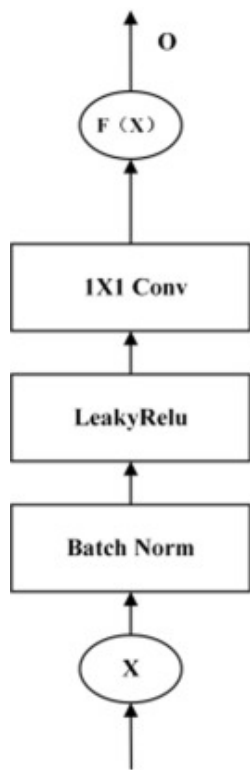
[Download: Download full-size image](#)

Fig. 5. Basic dense block architecture.

3.4. Transition block structure design

It is mainly a basic unit connecting two adjacent by SA, Dense Block and Residual Block by contact feature fusion, mainly used to reduce the feature map channel size. The basic architecture of Transition Block is shown in Fig. 6. From Fig. 6, we can see that the Transition Block consists of BatchNorm, LeakyRelu and 1x1conv in order.

The Transition layer can play the role of a compression model. Assuming that the number of feature map channels obtained from the front end of Transition is m , the Transition layer can generate θm features (through the convolution layer), where $\theta \in (0, 1]$ is the compression factor.



[Download: Download high-res image \(56KB\)](#)

[Download: Download full-size image](#)

Fig. 6. Transition block basic architecture.

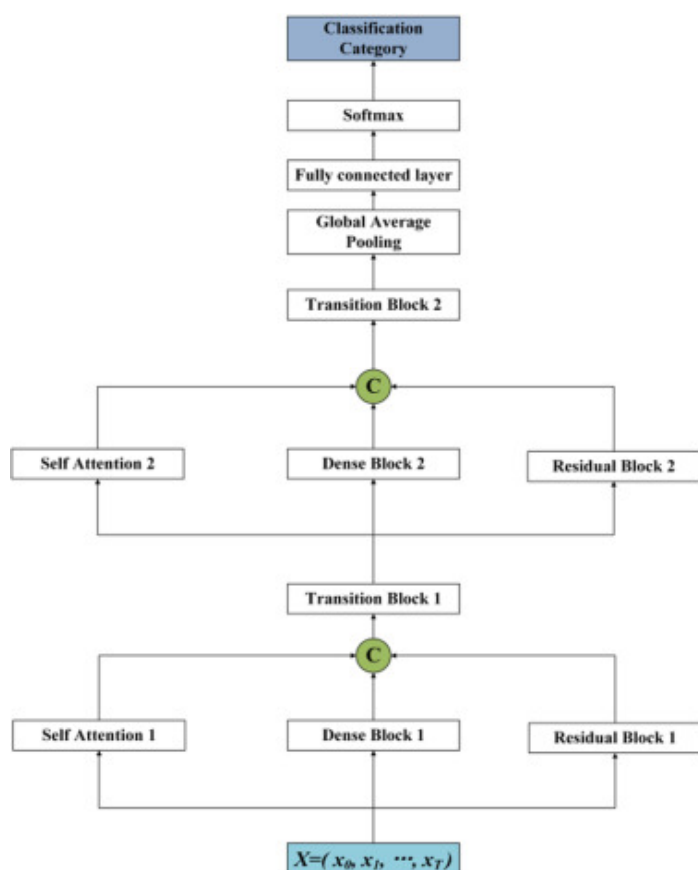
3.5. Design of the proposed SACDCNN

Based on the components described earlier, we propose SACDCNN. Its architecture design is shown in Fig. 7. As can be seen from Fig. 7, for the temporal information all went through Self Attention 1, Dense Block 1, and Residual Block 1 respectively for the first step of feature extraction. Then they were fused for the contact features. Furthermore, the features were compressed after Transition Block 1 to get the preliminary features. Then the initial features were extracted by Self Attention 2, Dense Block 2, and Residual Block 2, and they were fused with contact features. In addition, these features are compressed by Transition block 2 to obtain further features.

The new features are then obtained by global average pooling, which is chosen here because it has these advantages. Compared to the fully connected layer, using global average pooling is a more parsimonious choice of convolutional structure for establishing the relationship between feature maps and categories. The global averaging pooling layer does not require parameters to avoid overfitting at this layer. The global average pooling sums the spatial information and is more robust

to spatial variations in the input. The predicted classes are then passed through a fully connected layer and can be obtained after SoftMax.

The settings of this experiment for the SACDCNN structure parameters are shown in Table 2. All dropout rates are set to 0.05. From Table 2, the kernel size of the Causal Dilated Convolution is consistent, but its dilated factor and the number of filters is gradually increasing. This is to obtain a larger perceptual field from the convolution operation and thus extract the important features.



Download: [Download high-res image \(157KB\)](#)

Download: [Download full-size image](#)

Fig. 7. Architecture of SACDCNN.

Table 2. Structure parameters settings of SACDCNN.

Self Attention 1	Dense Block 1	Residual Block 1
The output is the same size as the input features	Conv1: Kernel size:8 Output Channel:64 dilation=1	Conv1: Kernel size:8 Output Channel:64 dilation=1
	Conv2: Kernel size:8 Output Channel:64 dilation=2	Conv2: Kernel size:8 Output Channel:64 dilation=2

Transition Block 1: $\theta = 0.5$ **Self Attention 2**

The output is the same size as the input features

Dense Block 2

Conv3: Kernel size: 8

Output Channel: 128

dilation=4

Conv4: Kernel size: 8

Output Channel: 128

dilation=8

Residual Block 2

Conv3: Kernel size: 8

Output Channel: 128

dilation=4

Conv4: Kernel size: 8

Output Channel: 128

dilation=8

Transition Block 2: $\theta = 0.5$ **Global Average Pooling**

Output of the fully connected layer: Number of classification categories

SoftMax

3.6. Network implementation

Given a time series training sample $T = \{(x_i, y_i)\}_{i=1}^N$. Considering that the SACDCNN network performs a classification task. To calculate the loss between the output and the corresponding labels, we used the most used cross-entropy loss function for prediction error. The formula is defined as follows.

$$Loss = -\frac{1}{N} \sum_{i=0}^{N-1} \sum_{k=0}^{K-1} y_i \log p_{i,k} \quad (16)$$

where N is the number of samples predicted, K is the number of classification categories and $p_{i,k}$ is the probability that the i th sample is predicted to be the k th label value.

Once the final optimization function of the SACDCNN is determined, the optimal parameters of the SACDCNN can be obtained by optimizing the loss function to search. The network is trained using the Adam optimizer to update the parameters. The samples are randomly disrupted and divided into batches. The training process is repeated until the losses converge to the desired value. The Parameter set W is updated as follows.

$$W \leftarrow W - \eta \left(\frac{\partial Loss_{Total}}{\partial W} \right) \quad (17)$$

where η is the learning rate.

The details of the pseudo-code for training SACDCNN are shown in Algorithm 1.

Algorithm 1. Training of SACDCNN.

Input: $T = \{(x_i, y_i)\}_{i=1}^N$, batch size n_b , learning rate η , Max_epoch

Initialize the parameters $\{W^{(l)}, b^{(l)}\}$ of SACDCNN.

For epoch=1: Max_epoch

 Calculate basic block $\hat{S}_i^{(j,L)}$

 Calculate temporal block $S_i^{(j,L)}$

 Calculate Resnet block RZ_i

 Calculate Densenet block DZ_i

 Calculate Self-Attention block SA_i

 Calculate total loss by Eq. (16)

 Calculate gradient and update parameters $W^{(l)}, b^{(l)}$

End

Output: Weights and biases $\{W^{(l)}, b^{(l)}\}$

Download: [Download high-res image \(243KB\)](#)

Download: [Download full-size image](#)

4. Case study

4.1. Experiment settings

Our proposed SACDCNN was tested on the same subset of 20 MTS obtained classification models (TCN, FCN, [MLP](#), Resnet, Twiesn, Encoder, CNN, Inception). The experimental environment is Ubuntu 17.10, Python 3.7 programming environment, TensorFlow and Pytorch [Deep Learning Framework](#), 64G RAM, GeForce RTX 2080Ti 11 GB. Details of all datasets are shown in [Table 3](#). The benchmark dataset taken for this experiment was derived from the UCR and UCI databases. You may view all data sets through <http://www.timeseriesclassification.com/index.php> and <https://archive.ics.uci.edu/ml/index.php>. The database of the original dataset has been divided between the training set and the test set, and we did not make any further changes.

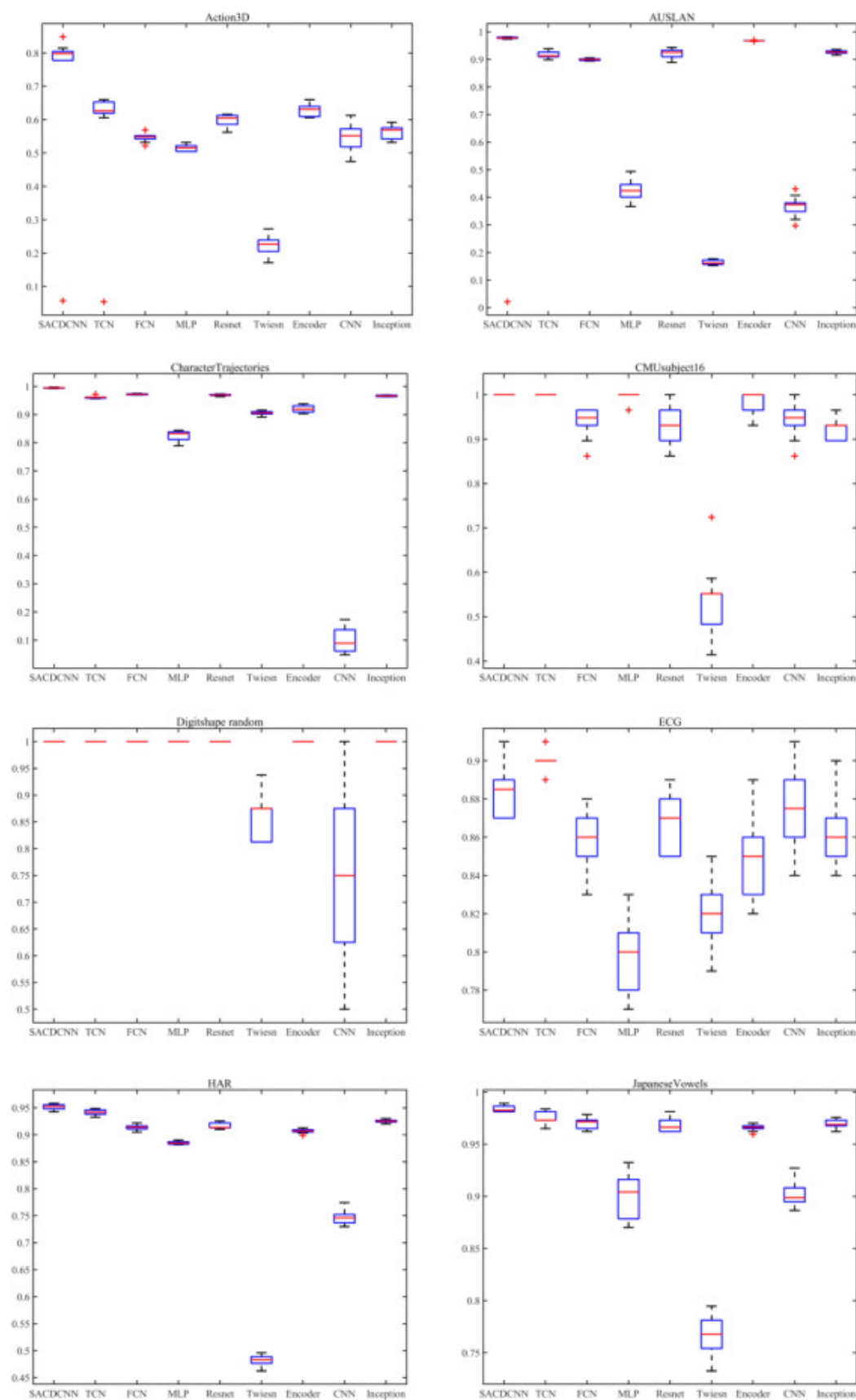
As can be seen from [Table 3](#), the most variables in the dataset are Action3D with 570, and the least number of variables is 2. The most extended sequence length is WalkvsRun with 1919. The shortest [time step](#) is 15. The training and test sets have the highest number of HAR, 7352 and 2947, respectively. Moreover, the most classification categories are AUSLAN with 95, and the least is 2.

Table 3. 20 benchmark MTS datasets.

Dataset	Number of variables	Max length	Train	Test	Classes
Action3D	570	100	270	297	20
AUSLAN	22	136	1140	1425	95
CharacterTrajectories	3	205	300	2558	20

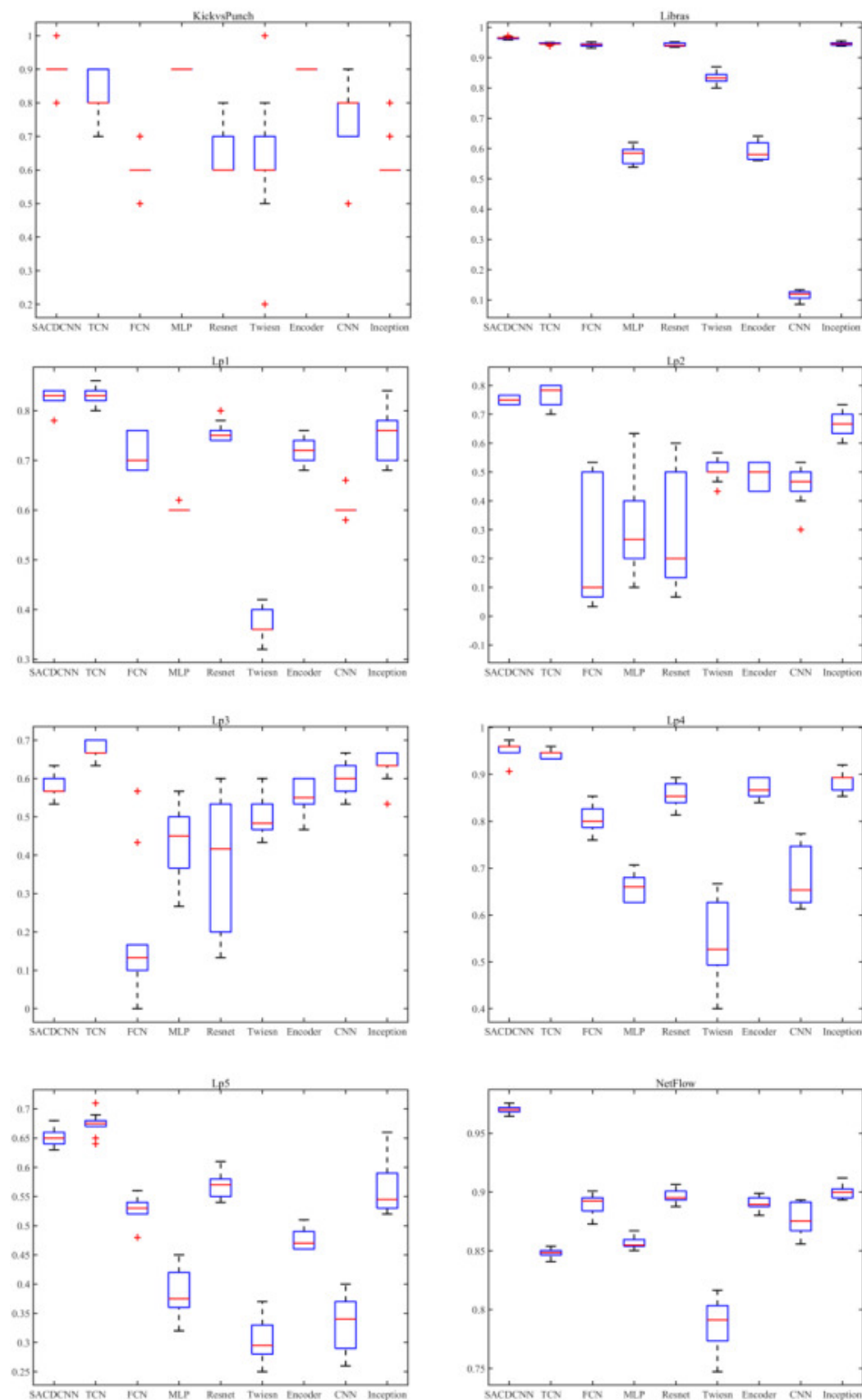
CMUsubject16	62	580	29	29	2
Digitshape random	2	97	24	16	4
ECG	2	152	100	100	2
HAR	9	128	7352	2947	6
JapaneseVowels	12	29	270	370	9
KickvsPunch	62	841	16	10	2
Libras	2	45	180	180	15
Lp1	6	15	38	50	4
Lp2	6	15	17	30	5
Lp3	6	15	17	30	4
Lp4	6	15	42	75	3
Lp5	6	15	64	100	5
NetFlow	4	994	803	534	2
Shapes random	2	97	18	12	3
UWave	3	315	200	4278	8
Wafer	6	198	298	896	2
WalkvsRun	62	1919	28	16	2

It is worth noting that since the original data set is not divided into validation sets, this experiment will randomly select 30% from the test set as the validation set and the remaining part as the new test set. The learning algorithm used for all models is the Adam optimizer. The batch of each model in this comparison experiment is set to 64, and the epoch is 500. According to the original literature, the learning rates of SACDCNN, TCN, FCN, MLP, Resnet, Twiesn, Encoder, CNN and Inception were set to 0.0001, 0.0001, 0.001, 0.001, 0.001, 0.0001, 0.00001, 0.001 and 0.0001, respectively. The details of the network structure of these comparison models are consistent with the original literature and will not be repeated for the sake of brevity.



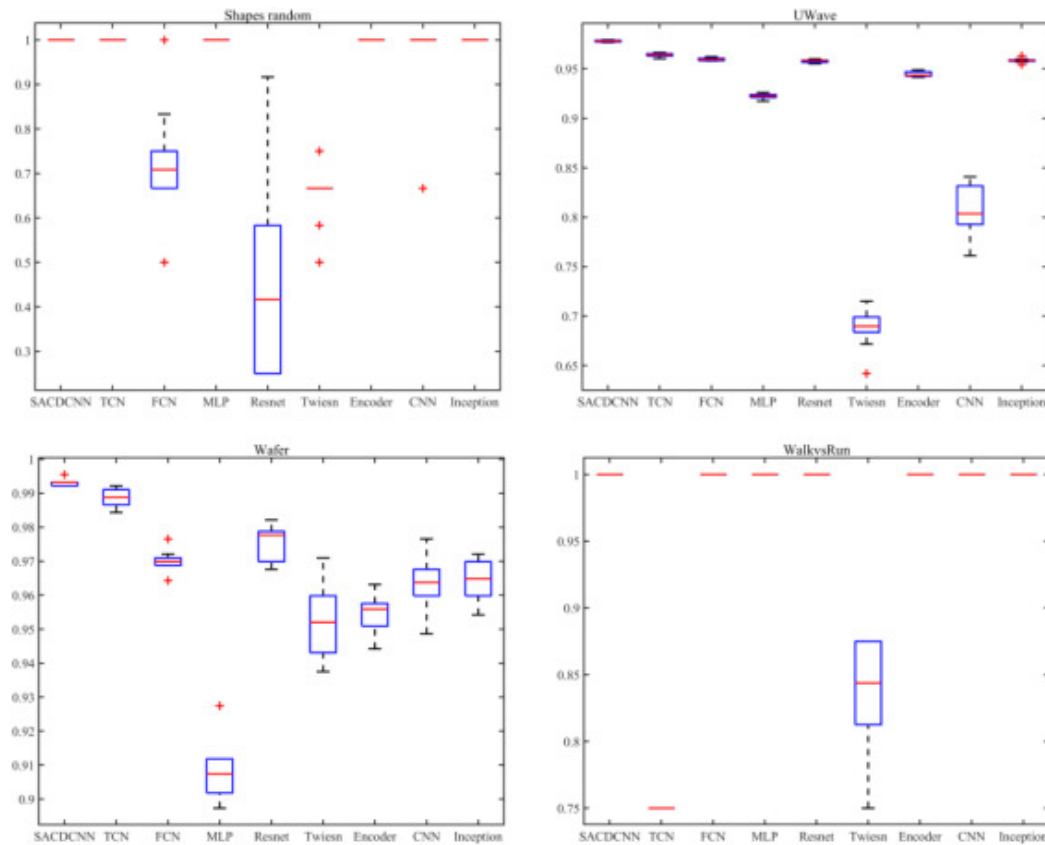
[Download: Download high-res image \(523KB\)](#)

[Download: Download full-size image](#)



[Download: Download high-res image \(540KB\)](#)

[Download: Download full-size image](#)



Download: [Download high-res image \(251KB\)](#)

Download: [Download full-size image](#)

Fig. 8. Numerical distribution of the classification accuracy of each model.

We further processed the accuracy obtained from 30 runs on the test set to obtain the average (Ave), standard deviation (Std) and maximum (Max) and minimum (Min) values and used them as evaluation metrics. To provide a simple and straightforward view of the stability of each model's performance on the test set, the widely used box-line plots were used. In addition, we performed pairwise post hoc analyses of nonparametric tests, and for this experiment, we chose the widely used Wilcoxon rank sum test (Harris and Hardin, 2013), whose significance evaluation index was set at 5%.

4.2. Experimental results and analysis

The test results for the 20 benchmark MTS datasets are shown in Table 4, where the bolded data are the optimal values. From Table 4, SACDCNN shows significant competitiveness and stability on 14 datasets. On CMUsubject16, SACDCNN and TCN show the best performance under all performance metrics. On Digitshape random, only Twiesn and CNN do not perform well. TCN shows the best performance on LP1, LP2 LP3, and LP5, followed by SACDCNN. On Shapes random, SACDCNN, TCN, MLP, Encoder, and Inception all show excellent performance. On WalkvsRun, only TCN and Twiesn perform poorly, while all others show the same excellent performance.

Table 4. Performance of the models on the benchmark multivariate data set.

Dataset	Criteria	SACDCNN	TCN	FCN	MLP	Resnet	Twiesn	Encoder	CNN	Inception
Action3D	Ave	0.8182	0.5758	0.5481	0.5162	0.5970	0.2253	0.6279	0.5471	0.5643
	Std	0.0144	0.1748	0.0139	0.0101	0.0178	0.0310	0.0164	0.0396	0.0197
	Max	0.8418	0.6599	0.5690	0.5320	0.6162	0.2727	0.6599	0.6128	0.5926
	Min	0.7980	0.0539	0.5219	0.5051	0.5623	0.1717	0.6061	0.4747	0.5320
AUSLAN	Ave	0.9782	0.9176	0.8993	0.4261	0.9211	0.1641	0.9674	0.3662	0.9264
	Std	0.0033	0.0121	0.0035	0.0335	0.0174	0.0082	0.0012	0.0367	0.0062
	Max	0.9832	0.9382	0.9053	0.4940	0.9432	0.1775	0.9698	0.4302	0.9361
	Min	0.9726	0.8982	0.8940	0.3670	0.8891	0.1530	0.9656	0.2975	0.9151
CharacterTrajectories	Ave	0.9948	0.9613	0.9720	0.8255	0.9692	0.9050	0.9197	0.0974	0.9665
	Std	0.0013	0.0040	0.0019	0.0178	0.0029	0.0074	0.0118	0.0429	0.0022
	Max	0.9973	0.9722	0.9750	0.8440	0.9734	0.9159	0.9382	0.1732	0.9695
	Min	0.9926	0.9570	0.9695	0.7897	0.9636	0.8913	0.9030	0.0477	0.9633
CMUsubject16	Ave	1.0000	1.0000	0.9379	0.9966	0.9310	0.5379	0.9862	0.9414	0.9207
	Std	0.0000	0.0000	0.0338	0.0103	0.0408	0.0804	0.0229	0.0379	0.0221
	Max	1.0000	1.0000	0.9655	1.0000	1.0000	0.7241	1.0000	1.0000	0.9655
	Min	1.0000	1.0000	0.8621	0.9655	0.8621	0.4138	0.9310	0.8621	0.8966
Digitshape random	Ave	1.0000	1.0000	1.0000	1.0000	1.0000	0.8625	1.0000	0.7438	1.0000
	Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0468	0.0000	0.1592	0.0000
	Max	1.0000	1.0000	1.0000	1.0000	1.0000	0.9375	1.0000	1.0000	1.0000
	Min	1.0000	1.0000	1.0000	1.0000	1.0000	0.8125	1.0000	0.5000	1.0000
ECG	Ave	0.8740	0.9010	0.8590	0.7980	0.8680	0.8200	0.8490	0.8750	0.8610
	Std	0.0162	0.0054	0.0158	0.0204	0.0133	0.0184	0.0212	0.0206	0.0170
	Max	0.9200	0.9100	0.8800	0.8300	0.8900	0.8500	0.8900	0.9100	0.9000
	Min	0.8600	0.8900	0.8300	0.7700	0.8500	0.7900	0.8200	0.8400	0.8400
HAR	Ave	0.9517	0.9415	0.9136	0.8850	0.9159	0.4815	0.9071	0.7460	0.9248
	Std	0.0048	0.0052	0.0043	0.0027	0.0051	0.0091	0.0038	0.0121	0.0031
	Max	0.9583	0.9488	0.9220	0.8904	0.9253	0.4961	0.9128	0.7743	0.9301
	Min	0.9423	0.9325	0.9050	0.8816	0.9101	0.4625	0.8992	0.7299	0.9199

JapaneseVowels	Ave	0.9868	0.9751	0.9703	0.8997	0.9686	0.7654	0.9657	0.9016	0.9689
	Std	0.0033	0.0055	0.0053	0.0195	0.0070	0.0199	0.0030	0.0112	0.0042
	Max	0.9946	0.9838	0.9784	0.9324	0.9811	0.7946	0.9703	0.9270	0.9757
	Min	0.9811	0.9649	0.9622	0.8703	0.9622	0.7324	0.9595	0.8865	0.9622
KickvsPunch	Ave	0.9000	0.8200	0.5900	0.9000	0.6500	0.6200	0.9000	0.7700	0.6300
	Std	0.0447	0.0600	0.0539	0.0000	0.0806	0.1939	0.0000	0.1100	0.0640
	Max	1.0000	0.9000	0.7000	0.9000	0.8000	1.0000	0.9000	0.9000	0.8000
	Min	0.8000	0.7000	0.5000	0.9000	0.6000	0.2000	0.9000	0.5000	0.6000
Libras	Ave	0.9651	0.9456	0.9429	0.5788	0.9422	0.8337	0.5911	0.1161	0.9460
	Std	0.0037	0.0031	0.0055	0.0254	0.0058	0.0192	0.0281	0.0140	0.0050
	Max	0.9726	0.9504	0.9521	0.6205	0.9521	0.8701	0.6410	0.1333	0.9556
	Min	0.9590	0.9385	0.9316	0.5385	0.9350	0.8000	0.5607	0.0855	0.9385
Lp1	Ave	0.8260	0.8300	0.7160	0.6040	0.7560	0.3720	0.7220	0.6040	0.7540
	Std	0.0180	0.0161	0.0332	0.0080	0.0196	0.0312	0.0275	0.0196	0.0474
	Max	0.8400	0.8600	0.7600	0.6200	0.8000	0.4200	0.7600	0.6600	0.8400
	Min	0.7800	0.8000	0.6800	0.6000	0.7400	0.3200	0.6800	0.5800	0.6800
Lp2	Ave	0.7500	0.7700	0.2267	0.3000	0.2767	0.5067	0.4833	0.4567	0.6700
	Std	0.0167	0.0348	0.2032	0.1498	0.1820	0.0359	0.0428	0.0651	0.0433
	Max	0.7667	0.8000	0.5333	0.6333	0.6000	0.5667	0.5333	0.5333	0.7333
	Min	0.7333	0.7000	0.0333	0.1000	0.0667	0.4333	0.4333	0.3000	0.6000
Lp3	Ave	0.5767	0.6733	0.1900	0.4400	0.4000	0.4967	0.5533	0.6000	0.6333
	Std	0.0260	0.0249	0.1633	0.0892	0.1653	0.0504	0.0452	0.0422	0.0394
	Max	0.6333	0.7000	0.5667	0.5667	0.6000	0.6000	0.6000	0.6667	0.6667
	Min	0.5333	0.6333	0.0000	0.2667	0.1333	0.4333	0.4667	0.5333	0.5333
Lp4	Ave	0.9560	0.9453	0.8053	0.6600	0.8547	0.5413	0.8707	0.6747	0.8853
	Std	0.0134	0.0093	0.0275	0.0268	0.0256	0.0827	0.0207	0.0609	0.0232
	Max	0.9733	0.9600	0.8533	0.7067	0.8933	0.6667	0.8933	0.7733	0.9200
	Min	0.9333	0.9333	0.7600	0.6267	0.8133	0.4000	0.8400	0.6133	0.8533
Lp5	Ave	0.6570	0.6740	0.5280	0.3820	0.5690	0.3050	0.4760	0.3330	0.5610

NetFlow	Std	0.0237	0.0185	0.0204	0.0363	0.0207	0.0347	0.0180	0.0427	0.0401
	Max	0.7000	0.7100	0.5600	0.4500	0.6100	0.3700	0.5100	0.4000	0.6600
	Min	0.6200	0.6400	0.4800	0.3200	0.5400	0.2500	0.4600	0.2600	0.5200
	Ave	0.9702	0.8479	0.8899	0.8564	0.8961	0.7860	0.8903	0.8764	0.9006
Shapes random	Std	0.0032	0.0035	0.0079	0.0052	0.0058	0.0204	0.0054	0.0136	0.0065
	Max	0.9757	0.8539	0.9007	0.8670	0.9064	0.8165	0.8989	0.8933	0.9120
	Min	0.9644	0.8408	0.8727	0.8502	0.8876	0.7472	0.8801	0.8558	0.8933
	Ave	1.0000	1.0000	0.7250	1.0000	0.4750	0.6583	1.0000	0.9333	1.0000
UWave	Std	0.0000	0.0000	0.1239	0.0000	0.2110	0.0692	0.0000	0.1333	0.0000
	Max	1.0000	1.0000	1.0000	1.0000	0.9167	0.7500	1.0000	1.0000	1.0000
	Min	1.0000	1.0000	0.5000	1.0000	0.2500	0.5000	1.0000	0.6667	1.0000
	Ave	0.9782	0.9640	0.9597	0.9224	0.9579	0.6872	0.9448	0.8070	0.9585
Wafer	Std	0.0015	0.0018	0.0012	0.0023	0.0013	0.0187	0.0024	0.0230	0.0021
	Max	0.9805	0.9665	0.9623	0.9260	0.9601	0.7150	0.9489	0.8409	0.9631
	Min	0.9763	0.9604	0.9581	0.9174	0.9553	0.6418	0.9417	0.7610	0.9545
	Ave	0.9932	0.9886	0.9696	0.9080	0.9752	0.9531	0.9547	0.9636	0.9646
WalkvsRun	Std	0.0009	0.0023	0.0034	0.0081	0.0051	0.0110	0.0050	0.0074	0.0058
	Max	0.9955	0.9922	0.9766	0.9275	0.9821	0.9710	0.9632	0.9766	0.9721
	Min	0.9922	0.9844	0.9643	0.8973	0.9676	0.9375	0.9442	0.9487	0.9542
	Ave	1.0000	0.7500	1.0000	1.0000	1.0000	0.8313	1.0000	1.0000	1.0000
	Std	0.0000	0.0000	0.0000	0.0000	0.0000	0.0488	0.0000	0.0000	0.0000
	Max	1.0000	0.7500	1.0000	1.0000	1.0000	0.8750	1.0000	1.0000	1.0000
	Min	1.0000	0.7500	1.0000	1.0000	1.0000	0.7500	1.0000	1.0000	1.0000

4.3. Model stability analysis

Although the standard deviation of the accuracy of each model on the benchmark dataset is given in Table 4, it still does not provide a visual representation of the numerical distribution of the accuracy of each model on the dataset. To represent the numerical distribution of all algorithms more intuitively, we use box plots to represent the accuracy distribution of each model after 30 independent tests. The reasons for box line plot as stability analysis of the model are as follows. It can

not only show the distribution, outliers, fluctuations, and stability of a single set of data, but also compare the differences in the distribution of different categories of data.

Fig. 8 shows the numerical distribution of the classification accuracy of each model on the test set. As can be seen in Fig. 8, SACDCNN has significant stability and excellent performance on the majority of datasets, with lower edge, upper edge, upper quartile, lower quartile, and median values consistently lower than other models under the same function. It is worth noting that only SACDCNN has the best distribution of values on NetFlow and UWave, while all other values have significant fluctuations.

4.4. Wilcoxon's rank-sum test analysis

To address the randomness factor that may affect the algorithm's performance, this experiment uses statistical tests to evaluate the performance difference between SACDCNN and other models. The widely used Wilcoxon rank sum test with a significance evaluation metric set at 5% was chosen for this experiment.

The p-values obtained from Wilcoxon's SACDCNN rank-sum test and other models are shown in Table 5, where the bolded words in the table are values greater than 5% or NaN. NaN indicates that both SACDCNN and the current model exhibit the best performance, i.e., 100% classification accuracy in each case. From Table 5, we can see that there is no significant statistical difference between SACDCNN and TCN, MLP and Encoder on CMUsubject16. There is a significant statistical difference between SACDCNN and Twiesn, CNN performance on Digitshape random. On KickvsPunch, there is no significant statistical difference between SACDCNN and MLP, Encoder. On LP1 and LP2, SACDCNN showed no significant statistical difference with TCN only. SACDCNN and Encoder, CNN showed no significant statistical difference on LP3. On Shapes random, SACDCNN showed only significant statistical difference with FCN, Resnet, Twiesn significant statistical difference. SACDCNN, TCN, and Twiesn showed significant statistical difference on WalkvsRun. On the other 12 datasets, SACDCNN and the compared models showed significant statistical difference.

Table 5. P-values obtained from Wilcoxon rank sum test for SACDCNN and other models.

Dataset	Index	SACDCNN VS.							
		TCN	FCN	MLP	Resnet	Twiesn	Encoder	CNN	Incepti
Action3D	p-value	2.14E-03	2.71E-03	2.71E-03	2.76E-03	2.77E-03	2.76E-03	2.77E-03	2.77E-03
AUSLAN	p-value	2.78E-03	2.79E-03	2.79E-03	2.79E-03	2.79E-03	2.67E-03	2.79E-03	2.78E-03
CharacterTrajectories	p-value	1.73E-04	1.76E-04	1.78E-04	1.78E-04	1.78E-04	1.78E-04	1.78E-04	1.76E-04

CMUsubject16	<i>p</i> -value	NaN	5.35E-05	3.68E-01	2.18E-04	5.94E-05	7.76E-02	2.11E-04	5.11E-04
Digitshape random	<i>p</i> -value	NaN	NaN	NaN	NaN	5.47E-05	NaN	2.15E-04	NaN
ECG	<i>p</i> -value	2.58E-03	4.90E-03	1.57E-04	4.40E-02	1.57E-04	1.86E-03	3.95E-01	7.40E-04
HAR	<i>p</i> -value	1.70E-03	1.83E-04	1.82E-04	1.82E-04	1.82E-04	1.83E-04	1.83E-04	1.79E-04
JapaneseVowels	<i>p</i> -value	8.27E-03	1.56E-04	1.60E-04	5.65E-04	1.60E-04	1.49E-04	1.60E-04	1.55E-04
KickvsPunch	<i>p</i> -value	7.48E-03	7.57E-05	1.00E-00	1.03E-04	1.85E-03	1.00E-00	2.40E-03	7.36E-04
Libras	<i>p</i> -value	1.49E-04	1.63E-04	1.68E-04	1.63E-04	1.69E-04	1.68E-04	1.64E-04	1.63E-04
Lp1	<i>p</i> -value	8.37E-01	1.43E-04	9.04E-05	2.11E-04	1.44E-04	1.48E-04	9.10E-05	2.05E-04
Lp2	<i>p</i> -value	1.31E-01	1.42E-04	1.44E-04	1.43E-04	1.33E-04	1.30E-04	1.36E-04	5.15E-04
Lp3	<i>p</i> -value	1.31E-04	4.14E-04	1.17E-03	1.22E-02	2.02E-03	3.45E-01	2.15E-01	5.18E-04
Lp4	<i>p</i> -value	2.23E-02	1.64E-04	1.62E-04	1.62E-04	1.67E-04	1.54E-04	1.62E-04	2.84E-04
Lp5	<i>p</i> -value	2.15E-02	1.69E-04	1.74E-04	1.71E-04	1.75E-04	1.57E-04	1.75E-04	1.25E-04
NetFlow	<i>p</i> -value	1.73E-04	1.76E-04	1.72E-04	1.75E-04	1.77E-04	1.73E-04	1.76E-04	1.75E-04
Shapes random	<i>p</i> -value	NaN	2.11E-04	NaN	6.02E-05	4.88E-05	NaN	1.67E-01	NaN
UWave	<i>p</i> -value	1.79E-04	1.78E-04	1.79E-04	1.75E-04	1.80E-04	1.78E-04	1.80E-04	1.76E-04
Wafer	<i>p</i> -value	2.16E-04	1.42E-04	1.42E-04	1.42E-04	1.46E-04	1.45E-04	1.45E-04	1.45E-04
WalkvsRun	<i>p</i> -value	1.59E-05	NaN	NaN	NaN	5.31E-05	NaN	NaN	NaN

4.5. Runtime and GPU memory analysis

This section provides an analysis of the average runtime and GPU memory consumed by the evaluation model on the benchmark dataset. The average runtime and the GPU memory occupied by the evaluated models are shown in Table 6 and Table 7, respectively, where the bolded font in the table is the best. From Table 6, SACDCNN has the least average runtime on the 12 data sets. CNN provided the fastest speed on 4 datasets. MLP had the best average runtime performance on 3 datasets. And Twiesn performs best on NetFlow only. As can be seen from Table 7, SACDCNN has the lowest average GPU memory usage on the 19 datasets. On Action3D, Twiesn has the best GPU memory usage with only 3.8966 GB.

Table 6. Average runtime of each evaluation model (Unit:s).

Dateset	SACDCNN	TCN	FCN	MLP	Resnet	Twiesn	Encoder	CNN	Inception
Action3D	359.39	395.00	98.87	256.62	193.67	290.24	210.11	76.94	258.59
AUSLAN	121.44	142.75	174.73	95.36	414.67	111.69	199.90	115.86	478.86
CharacterTrajectories	93.45	43.17	213.88	149.81	411.68	195.16	188.52	98.36	371.66
CMUsubject16	24.17	509.55	129.11	109.50	313.02	27.95	36.34	30.80	141.66
Digitshape random	20.96	29.87	78.73	42.64	168.96	23.83	37.76	28.12	162.48
ECG	27.55	36.74	75.87	41.41	158.87	43.34	47.75	36.12	155.49
HAR	638.22	929.26	607.88	327.29	1410.93	617.46	762.83	336.46	1718.76
JapaneseVowels	42.03	63.27	87.45	55.17	203.28	54.56	81.45	53.46	281.35
KickvsPunch	48.56	419.60	116.02	109.20	236.61	78.07	44.94	30.82	176.19
Libras	106.74	68.64	96.93	54.94	216.66	55.68	133.65	59.12	233.80
Lp1	29.68	38.93	127.56	61.37	296.23	43.83	48.87	30.64	196.38
Lp2	28.75	34.78	147.18	71.03	335.71	33.58	47.13	29.14	181.53
Lp3	28.40	38.61	129.74	70.53	272.86	43.02	36.73	16.87	89.08
Lp4	28.29	38.99	107.47	60.57	241.90	55.78	57.19	34.93	215.44
Lp5	29.10	39.00	105.08	57.35	237.05	43.66	51.47	25.77	152.12
NetFlow	78.96	184.93	128.44	68.69	318.25	47.45	162.46	91.70	361.83
Shapes random	38.83	59.71	102.26	54.21	265.00	63.91	47.24	28.99	162.47
UWave	86.20	113.73	182.88	121.04	372.10	76.26	268.60	131.22	421.14
Wafer	38.86	43.27	150.28	78.94	347.81	66.59	94.35	55.50	179.36

WalkvsRun	52.78	643.29	87.75	288.53	163.60	53.65	58.38	34.92	166.36
------------------	-------	--------	-------	--------	--------	-------	-------	--------------	--------

Table 7. Average GPU memory of each evaluation model (Unit:GB).

Dateset	SACDCNN	TCN	FCN	MLP	Resnet	Twiesn	Encoder	CNN	Inception
Action3D	6.9805	7.5020	9.2171	8.7398	9.8556	3.8966	9.6459	9.5352	8.9681
AUSLAN	2.1230	2.4746	4.3596	3.8824	4.9981	3.8692	4.7884	4.6778	4.1107
CharacterTrajectories	2.5898	3.5488	4.8264	4.3492	5.4649	3.9435	5.2552	5.1446	4.5775
CMUsubject16	3.5645	4.5430	5.8010	5.3238	6.4395	3.9376	6.2298	6.1192	5.5521
Digitshape random	1.6406	3.3965	3.8772	3.4000	4.5157	3.7911	4.3060	4.1954	3.6283
ECG	2.2070	3.5059	4.4436	3.9664	5.0821	3.9005	4.8724	4.7618	4.1947
HAR	2.2012	3.4902	4.4378	3.9605	5.0763	3.8849	4.8666	4.7559	4.1888
JapaneseVowels	1.6602	3.4043	3.8967	3.4195	4.5352	3.7989	4.3255	4.2149	3.6478
KickvsPunch	2.5527	4.5039	4.7893	4.3121	5.4278	3.8985	5.2181	5.1075	4.5404
Libras	1.8008	3.4160	4.0374	3.5601	4.6759	3.8107	4.4662	4.3555	3.7884
Lp1	1.5039	3.3887	3.7405	3.2633	4.3790	3.7833	4.1693	4.0587	3.4915
Lp2	1.5039	3.3672	3.7405	3.2633	4.3790	3.7618	4.1693	4.0587	3.4915
Lp3	1.5020	3.3672	3.7385	3.2613	4.3770	3.7618	4.1673	4.0567	3.4896
Lp4	1.4824	3.3926	3.7190	3.2418	4.3575	3.7872	4.1478	4.0372	3.4701
Lp5	1.5371	3.3965	3.7737	3.2965	4.4122	3.7911	4.2025	4.0919	3.5247
NetFlow	7.3594	6.1309	9.5960	9.1187	10.2345	4.5255	10.0248	9.9141	9.3470
Shapes random	1.7363	2.4004	3.9729	3.4957	4.6114	3.7950	4.4017	4.2911	3.7240
UWave	3.2207	5.6406	5.4573	4.9801	6.0958	4.0353	5.8861	5.7755	5.2083
Wafer	2.5938	3.5566	4.8303	4.3531	5.4688	3.9513	5.2591	5.1485	4.5814
WalkvsRun	8.8066	9.0078	10.7617	10.4355	11.6817	4.4025	11.4720	11.3614	10.7943

4.6. Convergence analysis of SACDCNN

In this section, convergence analysis is performed for SACDCNN on the training and validation sets.

The convergence metric of SACDCNN is the value of the cross-entropy loss function. The quantitative results of convergence metrics of SACDCNN are shown in Table 8. As can be seen from Table 8, the training loss of SACDCNN is always smaller than the validation loss on all data sets. The training accuracy of SACDCNN is 1.00 on each dataset, while the validation accuracy is variable. It should be added that this is because the experimental results are rounded, retaining two decimal places.

For a more intuitive analysis of convergence, the convergence curves of SACDCNN trained and validated on Libras and UWave are plotted. As shown in Fig. 9, where the horizontal axis is the number of iterations. From the figure, we can see that SACDCNN converges quickly on Libras and UWave. On Libras, the convergence of SACDCNN is relatively stable. On UWave, the convergence of SACDCNN is particularly fluctuating at some moments.

Table 8. Quantitative results of convergence metrics of SACDCNN.

Dateset	Train loss	Valid loss	Train acc	Valid acc
Action3D	1.00E-04	2.20E + 00	1.00	0.59
AUSLAN	1.20E-04	4.67E-01	1.00	0.89
CharacterTrajectories	3.04E-05	1.71E-01	1.00	0.97
CMUsubject16	2.14E-05	1.17E-01	1.00	0.93
Digitshape random	6.41E-05	2.38E-04	1.00	1.00
ECG	3.80E-05	8.91E-01	1.00	0.87
HAR	3.56E-07	1.21E + 00	1.00	0.92
JapaneseVowels	1.67E-05	1.66E-01	1.00	0.96
KickvsPunch	3.67E-05	6.26E-01	1.00	0.60
Libras	1.28E-04	3.00E-01	1.00	0.95
Lp1	7.86E-05	1.70E + 00	1.00	0.76
Lp2	7.02E-05	2.03E + 00	1.00	0.07
Lp3	5.33E-05	1.43E + 00	1.00	0.53
Lp4	6.57E-05	8.03E-01	1.00	0.85
Lp5	1.07E-04	2.31E + 00	1.00	0.61
NetFlow	6.88E-06	8.64E-01	1.00	0.90
Shapes random	3.84E-05	1.94E-01	1.00	0.92
UWave	7.97E-06	2.46E-01	1.00	0.96
Wafer	4.04E-05	1.01E-01	1.00	0.98

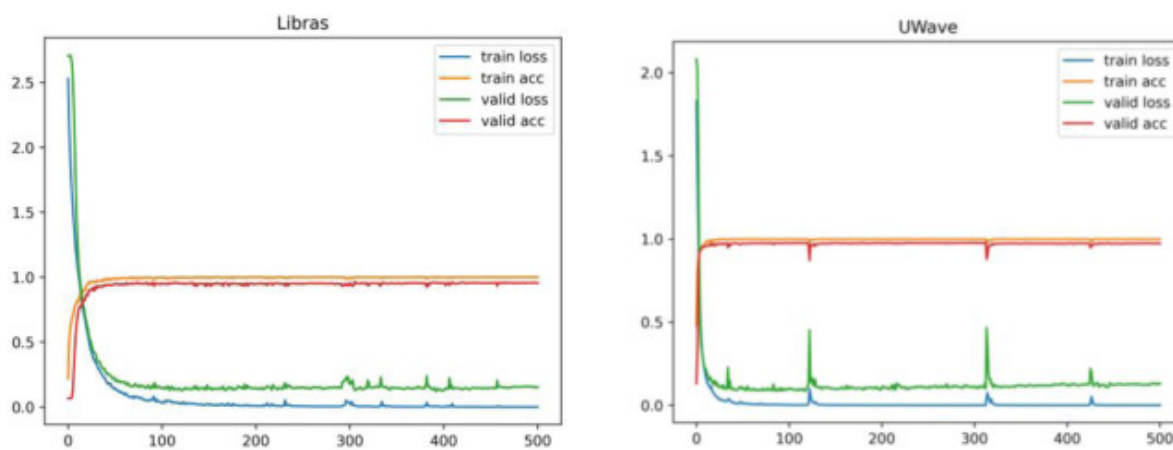
WalkvsRun

2.29E-05

2.39E-05

1.00

1.00



[Download: Download high-res image \(224KB\)](#)

[Download: Download full-size image](#)

Fig. 9. Convergence curves of SACDCNN trained and validated on Libras and UWave.

5. Petroleum engineering applications

5.1. SACDCNN-based framework for logging reservoir recognition

In response to the potential temporal correlation of oil logging data, the proposed SACDCNN is applied to solve the long-standing costly problem of logging reservoir recognition tasks. The framework consists of five steps, including missing data processing, feature selection, feature preprocessing, SACDCNN-based feature extraction, and logging reservoir recognition. Each step in this framework is described in detail as follows.

(1) Outliers and Missing Data Processing

The raw data usually has many missing data and outliers due to signal transmission interruption or acquisition equipment failure. A clustering-based approach is used to do outlier detection, which considers an object as an outlier based on clustering if the object does not firmly belong to any cluster.

The outliers are then considered as missing values and processed using the missing value processing method. For missing values, a polynomial interpolation method is used, which uses known data to fit a polynomial so that the existing data satisfies this polynomial and then solves for the missing values using the polynomial. We chose the most common method of polynomial interpolation, Lagrangian interpolation ([Berrut and Trefethen, 2004](#)).

(2) Noise treatment

It is generally considered that it is still quite difficult to recover all correct information in the data signal from incomplete or contaminated information, and the standard denoising methods are challenging to distinguish foreground and background information, so it is difficult to improve its information recovery accuracy and achieve the desired denoising effect. For this reason, a Robust Principal Component Analysis (RPCA) denoising method can be adopted from the sparse representation theory (Partridge and Jabri, 2000), which is based on a kind of pre-background separation idea, for the logging signal with valid foreground information and features, and further separates the target information from the foreground information by weakening the background information. This idea has been reflected in many algorithmic models in the field of speech and image processing.

(3) Normalization process

To overcome the influence of the dimensionality among the indicators, the influence of data inconsistency, and the convergence of model training, the sample information needs to be normalized, and the sample data is scaled to between [0,1]. The normalization of each logging attribute can be done by the following equation.

$$x' = \frac{x - x_{\min}}{x_{\max} - x_{\min}} \quad (18)$$

where $x \in [x_{\min}, x_{\max}]$, x_{\min} denotes the minimum value of the selected sample data and x_{\max} denotes the maximum value of the selected sample data.

(4) Data set partitioning

The data set is divided into a training set, a validation set, and a test set. The validation set is used to check the performance of the untrained model to avoid over-fitting and under-fitting. The training set is used to train the neural network model. The verification set is used to verify the validity of the model and select the model that gives the best result. Finally, the test set is used to test the final result of the model and evaluate the accuracy and error.

(5) Training and validation of SACDCNN model

The input data contains massive, multiscale, multi-type, multi-format, real-time, and dynamic characteristics, which poses a challenge for logging oil formation recognition in the petroleum industry. Fortunately, the SACDCNN-based feature extraction method not only effectively extracts the hidden information and long-term temporal relationships in the features, but also reduces the feature dimensionality of the input data.

The new feature information obtained from feature extraction is then used to train the SoftMax classification model, and the validation set is used for the tuning of the model parameters. After repeated iterations, the test set is input to the optimal SACDCNN model for logging oil formation recognition tasks in the oil industry, and the performance of the model is evaluated by several international standard evaluation metrics. In machine learning or deep learning, evaluation metrics

are a measure of how well or badly a model works. In the process of model evaluation, it is often necessary to use a variety of different metrics for evaluation. Among the many evaluation metrics, most of them can only one-sidedly reflect a part of the model performance, so we reasonably choose the accepted evaluation metrics, which include Accuracy, Precision, Recall, Root Mean Square Error (RMSE), and Mean Absolute Error (MAE).

5.2. Petroleum logging data sets

In this paper, we use real logging data from an oil and gas well in an oil field in China. For the convenience of differentiation, the oil wells are named well 1, and the gas wells are named well 2 in this experiment. The cores are homed by depth correction. Secondly, to reduce overfitting and underfitting of machine learning, data from well segments with dilation are removed. Well 1 has logging curves including CALI, K, TH, U, PE, NPHI, DEN, LLS, LLD, WQ, SP, DT, and GR. Well 2 has logging curves including SP, RXO, RT, RM, RI, RA4, RA3, NG, GR, CALI, and AC.

5.3. Evaluation metrics

In machine learning or deep learning, evaluation metrics are a measure of how well or badly a model works. In the process of model evaluation, it is often necessary to use a variety of different metrics for evaluation. Among the many evaluation metrics, most of them can only one-sidedly reflect a part of the model performance, so we reasonably choose the accepted evaluation metrics, which include Accuracy, Precision, Recall, Root Mean Square Error (RMSE), and Mean Absolute Error (MAE).

The formulas for these five performance metrics are defined as follows.

$$Accuracy = \frac{TP+TN}{TP+FP+TN+FN} \quad (19)$$

$$Precision = \frac{TP}{TP+FP} \quad (20)$$

$$Recall = \frac{TP}{TP+FN} \quad (21)$$

$$RMSE = \sqrt{\frac{1}{m} \sum_{i=1}^m (f(x)-y)^2} \quad (22)$$

$$MAE = \frac{1}{m} \sum_{i=1}^m |f(x)-y| \quad (23)$$

where TP represents true positive, TN represents true negative, FP represents false positive, and FN represents false negative, and $f(x)$ and y are the identified output value and expected output value, respectively.

Table 9. Petroleum layer recognition results for each model on well 1.

Subsets	Evaluation indicators	SACDCNN	TCN	FCN	MLP	Resnet	Twiesn	Encoder	CNN	Inception
Testing	Ave Accuracy	0.9417	0.9390	0.7398	0.7271	0.7896	0.7271	0.7575	0.9242	0.7752

set	Std	0.0019	0.0010	0.0067	0.0001	0.0396	0.0003	0.0069	0.0089	0.0271
	Max	0.9438	0.9396	0.7563	0.7271	0.8500	0.7271	0.7667	0.9333	0.8042
	Min	0.9375	0.9375	0.7313	0.7271	0.7354	0.7271	0.7438	0.9063	0.7333

Table 10. Best performance of each model for petroleum layer recognition on well 1.

Subsets	Evaluation indicators	SACDCNN	TCN	FCN	MLP	Resnet	Twiesn	Encoder	CNN	Inception
Testing set	Accuracy	0.9438	0.9396	0.7563	0.7271	0.8500	0.7271	0.7667	0.9333	0.8042
	Precision	0.9300	0.9215	0.6892	0.3635	0.8085	0.3635	0.7060	0.9253	0.7536
	Recall	0.9279	0.9275	0.6726	0.5000	0.8444	0.5000	0.7060	0.9041	0.7461
	RMSE	0.2372	0.2458	0.4937	0.5224	0.3873	0.5224	0.4830	0.2582	0.4425
	MAE	0.0563	0.0604	0.2438	0.2729	0.1500	0.2729	0.2333	0.0667	0.1958

Table 11. Gas layer recognition results for each model on well 2.

Subsets	Evaluation indicators	SACDCNN	TCN	FCN	MLP	Resnet	Twiesn	Encoder	CNN	Inception
Testing set	Average accuracy	0.9746	0.8804	0.7106	0.7792	0.7631	0.9077	0.8177	0.8313	0.7371
	Std	0.0089	0.0248	0.0067	0.0001	0.0372	0.0102	0.0131	0.0138	0.0298
	Max	0.9854	0.9292	0.7188	0.7792	0.8271	0.9208	0.8417	0.8521	0.8125
	Min	0.9604	0.8354	0.6979	0.7792	0.7125	0.8938	0.8000	0.8104	0.6979

5.4. Application results and analysis

The experimental environment and the parameter settings of the comparison models were kept the same as before. We conducted 30 independent comparison experiments for each model on each dataset. The experimental results are the performance evaluation metrics on the testing set, which are the evaluation of the generalization ability of the model and are the key factors in the final evaluation of the model performance.

For Well 1, Table 9 shows the oil layer recognition results of each model on Well 1. As seen from Table 9, SACDCNN showed significant superiority and competitiveness and reached 0.9417 in average

accuracy, its maximum accuracy was 0.9438, and its minimum accuracy was 0.9375. TCN took the second place. CNN took the third place. It is noteworthy that each performance index is the same for both MLP and Twiesn.

Table 10 shows the results of the best oil layer recognition for each model on well 1. As can be seen from Table 10, SACDCNN still maintains the best performance in each performance index, including 0.9438 in accuracy, 0.9300 in Precision, 0.9279 in Recall, 0.2372 in RMSE, and 0.0563 in MAE. The next best-performing distributions are TCN and CNN, which also show good performance. The worst performers are MLP and Twiesn.

For well 2, Table 11 shows the gas layer recognition results of each model on well 2. From Table 11, SACDCNN showed significant superiority and competitiveness and reached 0.9746 in average accuracy, its maximum accuracy was 0.9854 and its minimum accuracy was 0.9604. Twiesn won the second place. TCN won the third place. And FCN performed the worst in gas layer recognition.

Table 12 shows the best results of each model for gas layer recognition on well 2. From Table 12, SACDCNN still maintains the best performance in each performance index, including 0.9854 in accuracy, 0.9908 in precision, 0.9670 in Recall, 0.1208 in RMSE, and 0.0146 in MAE. The next best-performing distributions are TCN and Twiesn, which also show the next best performers are TCN and Twiesn. The worst performer is FCN.

Overall, SACDCNN shows significant superiority and competitiveness over other models in logging reservoir recognition tasks, and SACDCNN is very relevant for making decisions in logging interpretation, oil and gas exploration and development, and geoengineering evaluation.

Table 12. Best performance of each model for gas layer recognition on well 2.

Subsets	Evaluation indicators	SACDCNN	TCN	FCN	MLP	Resnet	Twiesn	Encoder	CNN	Inception
Testing set	Accuracy	0.9854	0.9292	0.7188	0.7792	0.8271	0.9208	0.8417	0.8521	0.8125
	Precision	0.9908	0.8917	0.7199	0.3896	0.7804	0.8682	0.7912	0.7865	0.7704
	Recall	0.9670	0.9072	0.8195	0.5000	0.8890	0.9458	0.8984	0.8510	0.8797
	RMSE	0.1208	0.2661	0.5303	0.4699	0.4158	0.2814	0.3979	0.3846	0.4330
	MAE	0.0146	0.0708	0.2813	0.2208	0.1729	0.0792	0.1583	0.1479	0.1875

6. Conclusion

In this paper, a novel deep learning model SACDCNN is proposed to address the problem that the existing deep learning models do not perform well on MTSC tasks. To extract the dependencies of

long-range time series in layer and to ensure that the performance needs to remain superior as the network deepens, Residual network structures and Dense network structures based on Causal Dilated Convolution are proposed. A Self-Attention mechanism that can extract the internal autocorrelations of time series features is also incorporated. Thirty independent comparison experiments were conducted on 20 benchmark UCR/UCI MTS datasets with TCN, FCN, MLP, Resnet, Twiesn, Encoder, CNN, and Inception. Experimental results show that SACDCNN exhibits significant superiority and stability in MTSC tasks and is a feasible, effective, and high-performance time series classification model.

It was also applied to petroleum engineering, aiming to solve the problem of the high cost of traditional logging reservoir interpretation in terms of human, material and financial resources, and was applied to one oil well and one gas well. The application results showed that SACDCNN showed significant superiority and competitiveness in these two wells, which overcame the shortcomings of traditional logging interpretation and also improved the success rate and efficiency of logging oil and gas exploration.

In our future work plan, we will continue to explore high-performance time series classification models and apply them to engineering fields such as oil and gas, construction, finance, and healthcare to provide new ideas and methods for engineering practice.

CRedit authorship contribution statement

Wenbiao Yang: Conceptualization, Methodology, Software, Writing – original draft, Writing – review & editing, Supervision, Project administration, Data curation, Visualization, Resources. **Kewen Xia:** Funding acquisition. **Zhaocheng Wang:** Formal analysis. **Shurui Fan:** Investigation. **Ling Li:** Validation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This work was supported by the National Natural Science Foundation of China (No. [42075129](#)), Hebei Province Natural Science Foundation, China (No. [E2021202179](#)), and Key Research and Development Project from Hebei Province, China (No. [19210404D](#), No. [20351802D](#), No. [21351803D](#)).

[Recommended articles](#)

Data availability

Data will be made available on request

References

- [Alawneh et al., 2021](#) Alawneh L., Alsarhan T., Al-Zinati M., Al-Ayyoub M., Jararweh Y., Lu H.
Enhancing human activity recognition using deep learning and time series augmented data
J. Ambient Intell. Humaniz Comput., 12 (2021), pp. 10565-10580, [10.1007/s12652-020-02865-4](#) ↗
[View in Scopus](#) ↗ [Google Scholar](#) ↗
- [Arican and Aydin, 2022](#) Arican E., Aydin T.
An RGB-D descriptor for object classification
Romanian J. Inf. Sci. Technol., 25 (2022), pp. 338-349
[View in Scopus](#) ↗ [Google Scholar](#) ↗
- [Ayhan and Samet, 2016](#) Ayhan S., Samet H.
Time series clustering of weather observations in predicting climb phase of aircraft trajectories
Proceedings of the 9th ACM SIGSPATIAL International Workshop on Computational Transportation Science, IWCTS '16, ACM Press, New York, New York, USA (2016), pp. 25-30, [10.1145/3003965.3003968](#) ↗
[View in Scopus](#) ↗ [Google Scholar](#) ↗
- [Bai et al., 2018](#) Bai S., Kolter J.Z., Koltun V.
An empirical evaluation of generic convolutional and recurrent networks for sequence modeling
(2018)
[Google Scholar](#) ↗
- [Baydogan and Runger, 2016](#) Baydogan M.G., Runger G.
Time series representation and similarity based on local autopatterns
Data Min. Knowl. Discov., 30 (2016), pp. 476-509, [10.1007/s10618-015-0425-y](#) ↗
[View in Scopus](#) ↗ [Google Scholar](#) ↗
- [Berrut and Trefethen, 2004](#) Berrut J.-P., Trefethen L.N.
Barycentric Lagrange interpolation
SIAM Rev., 46 (2004), pp. 501-517, [10.1137/S0036144502417715](#) ↗
[View in Scopus](#) ↗ [Google Scholar](#) ↗
- [Bhaskar et al., 2021](#) Bhaskar N., Suchetha M., Philip N.Y.
Time series classification-based correlational neural network with bidirectional LSTM for automated detection of kidney disease
IEEE Sens. J., 21 (2021), pp. 4811-4818, [10.1109/JSEN.2020.3028738](#) ↗
[View in Scopus](#) ↗ [Google Scholar](#) ↗

[Borlea et al., 2022](#) Borlea I.-D., Precup R.-E., Borlea A.-B.

Improvement of K-means cluster quality by post processing resulted clusters

Procedia Comput. Sci., 199 (2022), pp. 63-70, [10.1016/j.procs.2022.01.009](#) ↗



[View PDF](#) [View article](#) [Google Scholar](#) ↗

[Cuturi and Doucet, 2011](#) Cuturi M., Doucet A.

Autoregressive kernels for time series

(2011)

[Google Scholar](#) ↗

[Hao et al., 2023](#) Hao S., Wang Z., Alexander A.D., Yuan J., Zhang W.

MICOS: Mixed supervised contrastive learning for multivariate time series classification

Knowl. Based Syst., 260 (2023), Article 110158, [10.1016/j.knosys.2022.110158](#) ↗



[View PDF](#) [View article](#) [View in Scopus](#) ↗ [Google Scholar](#) ↗

[Harris and Hardin, 2013](#) Harris T., Hardin J.W.

Exact wilcoxon signed-rank and Wilcoxon Mann–Whitney ranksum tests

Stata J. Promot. Commun. Stat. Stata, 13 (2013), pp. 337-343, [10.1177/1536867X1301300208](#) ↗

[View in Scopus](#) ↗ [Google Scholar](#) ↗

[He et al., 2016](#) He K., Zhang X., Ren S., Sun J.

Deep residual learning for image recognition

Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society (2016), pp. 770-778, [10.1109/CVPR.2016.90](#) ↗

[Google Scholar](#) ↗

[Huang et al., 2017](#) Huang G., Liu Z., van der Maaten L., Weinberger K.Q.

Densely connected convolutional networks

Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017, Institute of Electrical and Electronics Engineers Inc (2017), pp. 1226-2269, [10.1109/CVPR.2017.243](#) ↗

[View in Scopus](#) ↗ [Google Scholar](#) ↗

[Ismail Fawaz et al., 2020](#) Ismail Fawaz H., Lucas B., Forestier G., Pelletier C., Schmidt D.F., Weber J., Webb G.I., Idoumghar L., Muller P.A., Petitjean F.

InceptionTime: Finding AlexNet for time series classification

Data Min. Knowl. Discov., 34 (2020), pp. 1936-1962, [10.1007/s10618-020-00710-y](#) ↗

[View in Scopus](#) ↗ [Google Scholar](#) ↗

[Karim et al., 2019](#) Karim F., Majumdar S., Darabi H., Harford S.

Multivariate LSTM-FCNs for time series classification

Neural Netw., 116 (2019), pp. 237-245, [10.1016/j.neunet.2019.04.014](#) ↗



[View PDF](#) [View article](#) [View in Scopus](#) ↗ [Google Scholar](#) ↗

[Lyu et al., 2023](#) Lyu H., Huang D., Li S., Ng W.W.Y., Ma Q.

Multiscale echo self-attention memory network for multivariate time series classification

Neurocomputing, 520 (2023), pp. 60-72, [10.1016/j.neucom.2022.11.066](#) ↗



[View PDF](#)

[View article](#)

[View in Scopus](#) ↗

[Google Scholar](#) ↗

[Maharaj and Alonso, 2014](#) Maharaj E.A., Alonso A.M.

Discriminant analysis of multivariate time series: Application to diagnosis based on ECG signals

Comput. Stat. Data Anal., 70 (2014), pp. 67-87, [10.1016/j.csda.2013.09.006](#) ↗



[View PDF](#)

[View article](#)

[View in Scopus](#) ↗

[Google Scholar](#) ↗

[Naqvi et al., 2020](#) Naqvi R.A., Arsalan M., Rehman A., Rehman A.U., Loh W.-K., Paul A.

Deep learning-based drivers emotion classification system in time series data for remote applications

Remote Sens. (Basel), 12 (587) (2020), [10.3390/rs12030587](#) ↗

[Google Scholar](#) ↗

[Orsenigo and Vercellis, 2010](#) Orsenigo C., Vercellis C.

Combining discrete SVM and fixed cardinality warping distances for multivariate time series classification

Pattern Recognit., 43 (2010), pp. 3787-3794, [10.1016/j.patcog.2010.06.005](#) ↗



[View PDF](#)

[View article](#)

[View in Scopus](#) ↗

[Google Scholar](#) ↗

[Partridge and Jabri, 2000](#) Partridge M., Jabri M.

Robust principal component analysis

Neural Networks for Signal Processing X. Proceedings of the 2000 IEEE Signal Processing Society Workshop (Cat. No. 00TH8501), IEEE (2000), pp. 289-298, [10.1109/NNSP.2000.889420](#) ↗

[View in Scopus](#) ↗

[Google Scholar](#) ↗

[Pei et al., 2018](#) Pei W., Dibeklioglu H., Tax D.M.J., van der Maaten L.

Multivariate time-series classification using the hidden-unit logistic model

IEEE Trans. Neural Netw. Learn. Syst., 29 (2018), pp. 920-931, [10.1109/TNNLS.2017.2651018](#) ↗

[View in Scopus](#) ↗

[Google Scholar](#) ↗

[Serrà et al., 2018](#) Serrà J., Pascual S., Karatzoglou A.

Towards a universal neural network encoder for time series

(2018)

[Google Scholar](#) ↗

[Seto et al., 2015](#) Seto S., Zhang W., Zhou Y.

Multivariate time series classification using dynamic time warping template selection for human activity recognition

2015 IEEE Symposium Series on Computational Intelligence, IEEE (2015), pp. 1399-1406,
[10.1109/SSCI.2015.199](#) ↗

[View in Scopus](#) ↗ [Google Scholar](#) ↗

[Singh et al., 2017](#) Singh A., Deep K., Grover P.

A novel approach to accelerate calibration process of a k -nearest neighbours classifier using GPU

J. Parall. Distrib. Comput., 104 (2017), pp. 114-129, [10.1016/j.jpdc.2017.01.003](#) ↗



[View PDF](#) [View article](#) [View in Scopus](#) ↗ [Google Scholar](#) ↗

[Srivastava et al., 2015](#) Srivastava R.K., Greff K., Schmidhuber J.

Highway networks

(2015)

[Google Scholar](#) ↗

[Tanisaro and Heidemann, 2017](#) Tanisaro P., Heidemann G.

Time series classification using time warping invariant echo state networks

Proceedings - 2016 15th IEEE International Conference on Machine Learning and Applications, ICMLA 2016, Institute of Electrical and Electronics Engineers Inc (2017), pp. 831-836, [10.1109/ICMLA.2016.166](#) ↗

[Google Scholar](#) ↗

[Tuncel and Baydogan, 2018](#) Tuncel K.S., Baydogan M.G.

Autoregressive forests for multivariate time series modeling

Pattern Recognit., 73 (2018), pp. 202-215, [10.1016/j.patcog.2017.08.016](#) ↗



[View PDF](#) [View article](#) [View in Scopus](#) ↗ [Google Scholar](#) ↗

[Vaswani et al., 2017](#) Vaswani A., Brain G., Shazeer N., Parmar N., Uszkoreit J., Jones L., Gomez A.N., Kaiser Ł., Polosukhin I.

Attention Is All You Need

Adv. Neural Inf. Process. Syst. (2017)

[Google Scholar](#) ↗

[Wang et al., 2016](#) Wang Z., Yan W., Oates T.

Time series classification from scratch with deep neural networks: A strong baseline

(2016)

[Google Scholar](#) ↗

[Wistuba et al., 2015](#) Wistuba M., Grabocka J., Schmidt-Thieme L.

Ultra-fast shapelets for time series classification

(2015)

[Google Scholar](#) ↗

[Wu et al., 2022](#) Wu Y., Lian C., Zeng Z., Xu B., Su Y.

An aggregated convolutional transformer based on slices and channels for multivariate time series classification

IEEE Trans. Emerg. Top. Comput. Intell. (2022), pp. 1-12, [10.1109/TETCI.2022.3210992](#) ↗



[View PDF](#) [View article](#) [Google Scholar](#) ↗

[Yang et al., 2022](#) Yang C., Wang X., Yao L., Long G., Jiang J., Xu G.

Attentional gated Res2Net for multivariate time series classification

Neural Process. Lett. (2022), [10.1007/s11063-022-10944-0](#) ↗

[Google Scholar](#) ↗

[Zhai et al., 2016](#) Zhai G., Hu J., Zhao W., Zou C.

History, achievements and significance of scientific exploration wells: For the 30 th anniversary of the scientific exploration well program

Petrol. Explor. Dev., 43 (2016), pp. 167-181, [10.1016/S1876-3804\(16\)30021-0](#) ↗



[View PDF](#) [View article](#) [View in Scopus](#) ↗ [Google Scholar](#) ↗

[Zhao et al., 2017](#) Zhao B., Lu H., Chen S., Liu J., Wu D.

Convolutional neural networks for time series classification

J. Syst. Eng. Electron., 28 (2017), pp. 162-169, [10.21629/JSEE.2017.01.18](#) ↗

[View in Scopus](#) ↗ [Google Scholar](#) ↗

[Zheng et al., 2014](#) Zheng Y., Liu Q., Chen E., Ge Y., Zhao J.L.

Time series classification using multi-channels deep convolutional neural networks

(2014), pp. 298-310, [10.1007/978-3-319-08010-9_33](#) ↗

[View in Scopus](#) ↗ [Google Scholar](#) ↗

[Zheng et al., 2016](#) Zheng Y., Liu Q., Chen E., Ge Y., Zhao J.L.

Exploiting multi-channels deep convolutional neural networks for multivariate time series classification

Front Comput Sci, 10 (2016), pp. 96-112, [10.1007/s11704-015-4478-2](#) ↗

[View in Scopus](#) ↗ [Google Scholar](#) ↗

Cited by (5)

[Acoustic data detection in large-scale emergency vehicle sirens and road noise dataset](#)

2024, Expert Systems with Applications

[Show abstract](#) ✓

[Research on a novel photovoltaic power forecasting model based on parallel long and short-term time series network](#)

2024, Energy

[Show abstract](#) ✓

[Dual attention-based deep learning for construction equipment activity recognition considering transition activities and imbalanced dataset](#)

2024, Automation in Construction

[Show abstract](#) ✓

[Causal Spatio-Temporal Graph Foresting Against Confounding Bias ↗](#)

2024, Research Square

[Dilated convolution for enhanced extractive summarization: A GAN-based approach with BERT word embedding ↗](#)

2024, Journal of Intelligent and Fuzzy Systems

[View Abstract](#)

© 2023 Elsevier Ltd. All rights reserved.

All content on this site: Copyright © 2024 Elsevier B.V., its licensors, and contributors. All rights are reserved, including those for text and data mining, AI training, and similar technologies. For all open access content, the Creative Commons licensing terms apply.