*Computers, Materials & Continua* CMC, vol.65, no.1, pp.543-561, 2020

# Accurate Multi-Scale Feature Fusion CNN for Time Series Classification in Smart Factory

**Xiaorui Shao[1], Chang Soo Kim[1,*] and Dae Geun Kim[2]**

**Abstract:** Time series classification (TSC) has attracted various attention in the community of machine learning and data mining and has many successful applications such as fault detection and product identification in the process of building a smart factory. However, it is still challenging for the efficiency and accuracy of classification due to complexity, multi-dimension of time series. This paper presents a new approach for time series classification based on convolutional neural networks (CNN). The proposed method contains three parts: short-time gap feature extraction, multi-scale local feature learning, and global feature learning. In the process of short-time gap feature extraction, large kernel filters are employed to extract the features within the short-time gap from the raw time series. Then, a multi-scale feature extraction technique is applied in the process of multi-scale local feature learning to obtain detailed representations. The global convolution operation with giant stride is to obtain a robust and global feature representation. The comprehension features used for classifying are a fusion of short time gap feature representations, local multi-scale feature representations, and global feature representations. To test the efficiency of the proposed method named multi-scale feature fusion convolutional neural networks (MSFFCNN), we designed, trained MSFFCNN on some public sensors, device, and simulated control time series data sets. The comparative studies indicate our proposed MSFFCNN outperforms other alternatives, and we also provided a detailed analysis of the proposed MSFFCNN.

**Keywords:** Time Series Classifications (TSC), smart factory, Convolutional Neural Networks (CNN).

## 1 Introduction

Time series classification (TSC) is one of the critical factors for implementing smart factories in industry 4.0 due to many time series generated from the process of global production every day and everywhere, such as vibration signals and all kinds of sensor data: humidity sensor data, speed sensor data etc. All those data generated from machines react to the status of the machine or surroundings. By predicting the future status of machines and

[1] Department of Information Systems, Pukyong National University, Busan, 608737, South Korea.

[2] Korea Dyeing & Finishing Technology Institute, Busan, 608737, South Korea.

[*] Corresponding Author: Chang Soo Kim. Email: cskim@pknu.ac.kr.

Received: 20 April 2020; Accepted: 13 May 2020.

surroundings, decision-makers could make a reasonable adjustment in advance to avoid failure and downtime. Therefore, it enables the company and factory to increase production efficiency and save production costs. More importantly, ensure personal safety. Developing an accurate approach for TSC is the key to reach this achievement.

TSC consists of distance-based methods, feature-based methods [Xing, Pei and Keogh (2010); Zheng, Liu, Chen et al. (2014); Cui, Chen and Chen (2016)], and machine learning-based methods. Distance-based methods such as k-nearest neighbor (KNN) and support vector machine (SVM) could be used for TSC directly on raw time series by calculating the similarity or dissimilarity between two time-sequences. The measurement of calculating the similarity is defining the distance function, such as Manhattan distance, Euclidean distance, and maximum distance etc. However, Both KNN and SVM [Chen, Xu, Zuo et al. (2019)] require equal length and are sensitive to the dimension of the time series. To overcome the above shortcomings, Batista et al. [Batista, Wang and Keogh (2011)] proposed the Dynamic Time Warping (DTW) to perform time series classification. Additionally, combining the KNN and DWT can effectively improve prediction accuracy [Chotirat and Eamonn (2005)]. It is still problematic that DWT requires too many time and computing resources [Zheng, Liu, Chen et al. (2014)].

The key idea of feature-based methods for TSC is capturing most representations of raw time series. Some statistical components such as mean, standard deviation, maximum value, minimum value, skewness etc. have been applied as statistical-domain features for TSC [Lei and Wu (2020)]. Meanwhile, the lower-dimensional reshaped time series has been employed as a feature representation for TSC. Principle component analysis (PCA) was employed for TSC [Li (2015); Cao, Tian and Bai (2015)] due to it has an excellent dimensionality reduction capacity. To obtain rich and robust feature representations, transformation-based methods have been proposed for time series classification such as Fourier transform (FT), Fast Fourier transform (FFT) and wavelet transform (WT) [Hendrik (2008); Zhang, Ho, Lin et al. (2006)]. They transform the raw time series from time-domain into frequency-domain to find the strong, and novel feature expression for accurately classifying time series. The useful information of the transformed signal is highly concentrated on the low-frequency part, but the noise on the high-frequency part. The transformation as mentioned above consists of discrete and continue transforms (DT and CT). Generally, CT requires more computing resources and time than DT. After feature selection, applying one classifier such as logistic and SVM to classify the time series. Recently, another feature-based method shapelet has proven that it is powerful for TSC [Arathi and Govardhan (2015); Hills, Lines, Baranauskas et al. (2014); Ahn and Lee (2018)] and became popular. The above methods only use one single model for TSC may lose some critical information. Therefore, some ensemble approaches combine multiple classifiers have been proposed for time series classification and have achieved excellent performance [Wang, Yan and Oates (2017)]. For instance, the Elastic Ensemble (PROP) [Lines and Bagnall (2015)] combines 11 classifiers using elastic distance measures in a weighted ensemble scheme; The flat collective of transform-based ensembles (COTE) combines 35 different classifiers by extracting features from the time-frequency domain [Bagnall, Lines, Hills et al. (2015)]. However, all those methods need crafted feature selection and are time-consumption when having massive data, respectively.

Recently, the machine learning-based method for TSC has extracted various attention. Tradition machine learning methods contain SVM, decision tree, and random forest (RF). Although they can implement TSC without feature selection engineering, the performance is still deficient, which cannot satisfy our needs. Fortunately, the new deep learning-based method gives us a new choice. Notably, the convolutional neural network (CNN) has been a hot topic for TSC due to its black-box and dominant feature extraction characteristics. In CNN, extracted deep and robust features are fed into classifier automatically, that is, feature selection and classifying are integrated into one single framework for TSC. Follows are some CNN-based methods which already achieved great success in the domain of TSC. i.e., Cui et al. [Cui, Chen and Chen (2016)] proposed a multi-scale CNN (MCNN) for TSC and verified its excellent performance through 44 UCR time series archives. However, we still need to execute some extra transforms to obtain multi-scale feature representations. To avoid extra operations simultaneously keeping high performance, Wang et al. [Wang, Yan and Oates (2017)] developed three deep learning-based methods: Fully CNN (FCNN), deep multilayer perceptron (MLP), and the residual networks (ResNet). They evaluated and analyzed those three methods on the same benchmark datasets to Cui's paper [Cui, Chen and Chen (2016)], the comparative experiments indicate the premium performance of FCNN. Zhao et al. [Zhao, Lu, Chen et al. (2017)] applied a classic CNN architecture for TSC and tested on UCR and simulated data sets. The biggest challenge of TSC using UCR achieves [Chen, Keogh, Hu at al. (2016)] is that training sets are much less. However, most of CNN architectures need to train the model with massive data. Cui et al. [Cui, Chen and Chen (2016)] proposed a sliding window (SW) data augmentation technology to generate more data sets. Besides, Guennec et al. [Guennec, Malinowski and Tavenard (2016)] employed a window warping (WW) method for data augmentation and compared with SW. The above methods only focused on univariate TSC (UTSC). As a matter of factor, time series that occurred in real-life may be multivariate. To deal with multivariate time series, Zheng et al. [Zheng, Liu, Chen et al. (2014)] proposed a multi-channel deep CNN (MCDCNN) for MTSC. Two channels were adopted to extract features in his paper, and they treated one-source time series as one channel to extract represented region individually. Extracted features are combined by one full connection layer. Liu et al. designed a multivariate CNN (MVCNN) for fault detection on prognostics and health management (PHM) data set [Liu, Hsaio and Tu (2019)]. In his paper, multi-source time series is transformed into three-dimensional (3-D) tensor as the input and then adopted MVCNN with four stages to capture the rich features for MTSC. Motivated by MCNN [Cui, Chen and Chen (2016)], Jiang et al. proposed [Jiang, He, Yan et al. (2018)] multi-scale CNN (MSCNN) for fault diagnosis of wind turbine gearbox, in which, the authors adopted three scales of the mean of each time series at different time gap for feature extraction. However, it did not test in the case of using a lack of data to train the model. Yazdanbakhsh et al. [Yazdanbakhsh and Stick (2019)] proposed a dilated convolutional neural network for MTC and validated its effectiveness on two human activity recognition time series (WISDM v1. 1 [Kwapisz, Weiss, Moore et al. (2011)] and WISDM v. 2 [Lockhart, Weiss, Xue et al. (2011)]). However, the accuracy is still lower than some traditional feature-based methods. The main related works for TSC using CNN-based methods as summarized in Tab. 1.

**Table 1:** Main related CNN-based methods for TSC

| Paper | Classifier | Dataset | Type |
|---|---|---|---|
| [Zheng, Liu, Chen et al. (2014)] | MCDCNN | PAMAP2/ BIDMC | MTSC |
| [Cui, Chen and Chen (2016)] | MCNN | UCR | UTSC |
| [Guennec, Malinowski and Tavenard (2016)] | t-leNet-DM+SVM | UCR | UTSC |
| [Wang, Yan and Oates (2017)] | FCNN, MLP, ResNet | UCR | UTSC |
| [Zhao, Lu, Chen et al. (2017)] | Classic CNN (CCNN) | UCR/ simulated data | Both |
| [Jiang, He, Yan et al. (2018)] | MSCNN | - | UTSC |
| [Liu, Hsaio and Tu (2019)] | MVCNN | PHM2015 | MTSC |
| [Yazdanbakhsh and Stick (2019)] | Dilated CNN | WISDM v1.1/ WISDM v. 2 | MTSC |

From Tab. 1, we can see most of the CNN-based research focused on one type of TSC, only Zhao et al. [Zhao, Lu, Chen et al. (2017)] employed classical CNN to do both them. Another drawback of the current CNN-based methods for TSC we discussed above is they still need other transformation or preprocessing operations. Moreover, some of them cannot deal with fewer data, as well. Motived by this, we designed, trained, and evaluated MSFFCNN to deal with those issues without any handcrafted feature engineering. We double that the reason for the above issues existing is cannot mine rich, robust, and detailed key representations of raw time series. Therefore, in our proposed model, we adopted a cascading structure to capture abundant feature maps. The main contributions of the manuscript are summarized as follows:

- To the best of our understanding, a few types of research focused on using one CNN-based model for both UTSC and MTSC. This paper addresses this issue with MSFFCNN.
- The cascading structure of MSFFCNN is detailly designed, trained, and verified on both univariate and multivariate time series. The comparative studies indicate our proposed method outperforms other excellent methods without special preprocessing operations.
- The feature learning capacity of the proposed method is analyzed and learned inner feature map is visualized.

The rest of the paper is arranged as follows. Section 2 gives the problem definition of TSC. Section 3 introduces CNN for TSC and depicts the proposed framework. Detailed experiment verifications are carried out in Section 4. In Section 5, we discuss the effectiveness of the proposed MSFFCNN. At last, we present the conclusions of this manuscript.

## 2 Problem definition

The TSC problem is to predict the label of the time series, which could be subdivided into UTSC and MTSC according to the dimensions of time series. The univariate time series in smart factory mainly are vibration signals could be expressed as a sequence of real-valued data points at different timestamps, which could be written as Eq. (1). Where

$n$ is the length of timestamps, $t_i$ denotes the $i^{th}$ data point of vibration signal $UT$. We give the detailed definition of two types of TSC problems as follows.

$$UT = \{t_1, t_2, \cdots, t_i, \cdots, t_n\} \quad (1)$$

Definition 1: The UTSC problem is considered as a vibration signal regarding a label that could be formalized as $UD_i = \{(UT_i, L_i) | UT_i \in UT \ L_i \in Z^*\}$. Where $UD_i$ is a complete data sample including vibration signal, and regarding label $L_i$ must be a positive integer, the number of $L_i$ depends on how many statuses it has in a real-case production environment. The whole data set is formalized as $UD = \{UD_1, UD_2, UD_3, \dots, UD_i, \dots, UD_n\}$, where $n$ is samples of data set.

Definition 2: The multivariate time series is a set of univariate time series with the same timestamps that can be detonated as Eq. (2). where $m$ is the number of univariate time series. Empathy, the MTSC problem could be formalized as $MD_i = \{(MT_i, L_i) | MT_i \in MT \ L_i \in Z^*\}$. The whole data set could be written as $MD = \{MD_1, MD_2, MD_3, \dots, MD_i, \dots, MD_n\}$. In this paper, we will apply UCR data sets for UTSC verification and real-life multivariate time series for MTSC verification.

$$MT = \{UT_1, UT_2, \cdots, UT_i, \cdots, UT_m\} \quad (2)$$

## 3 Methods

This paper proposed a novel MSFFCNN model to solve the TSC problem without any handcrafted feature engineering operations. The following two subsections give the related pre-knowledge of CNN and a detailed description of the proposed MSFFCNN.

### *3.1 CNN*

CNN is proposed by Lecun et al. [Lecun, Bengio and Hinton (2015)], it is a typical feedforward neural network and mainly employed for image classification, object detection. The standard CNN consists of two critical components: convolutional layer and pooling layer. Those two layers alternatively occurred in the CNN structure to extract rich feature maps within one sparse expression. The convolutional layer has properties of weights sharing, transformation, and scaling invariance. Consequently, it could extract robust feature representations. The process of convolution, as shown in Eq. (3). Where $x_i$ is $i^{th}$ input data points in the range of $j$ input values $M_j$, $y_i$ is the $j$ feature maps after convolution operation with $j$ filters $k_{ij}$, $b_j$ is $j$ basis of each feature map.

$$y_j = f(\sum_{i \in M_j} x_i \times k_{ij} + b_j) \quad (3)$$

After convolution operation, the convoluted data are processed with one activation function. This will make some data points active randomly to achieve the sparse representation. One of the most popular activation functions is the Rectified Linear Unit (ReLU), which enabled a nonlinear expression of input signals to enhance the representation ability, as shown in Eq. (4).

$$a_j = \max(0, y_j) \quad (4)$$

The pooling layer is used to reduce the dimension of features and speed up the convergence of the networks, which has three types of pooling down sample methods: Maximum pooling, minimum pooling, and average pooling. We give the maximum

pooling operation as follows:

$$p_j = max(q_j) \tag{5}$$

where $p_j$ is the output of maximum value among the obtained feature maps $q_j$ from the preceding layer; usually, the convolutional and pooling operations alternatively occurred in the CNN model to extract the deep, abstract, and global feature expressions [Peng and Marculescu (2015); Chen, Li and Sanchez (2015)]. CNN can handle well with one dimensional (1-D) signals [Liu, Yang, Lv et al. (2019)], two dimensional (2-D) images, and three dimensional (3-D) videos [Arif, Wang, Fei et al. (2019)]. For the TSC problem, we mainly apply CNN to deal with 1-D vibration signals.

### *3.2 Proposed deep model*

The architecture of proposed MSFFCNN for TSC consists of four parts: short-time gap feature learning, multi-scale feature learning, global feature learning and feature fusion, and output, as shown in Fig. 1. We concatenate multiple UTSC cells (marked with a red box in Fig. 1) for MTSC. Furthermore, different from the image classification problem, the input of the image classification problem is a two-dimension (2-D) image. The input of designed MSFFCNN is a one-dimension (1-D) time series. It learns feature through one raw time series for UTSC, and by combining the feature of multiple individual univariate time series for MTSC. The more detailed description of them will be depicted in the following subsections. We will take an example with UTSC to explain the workflow of MSFFCNN.

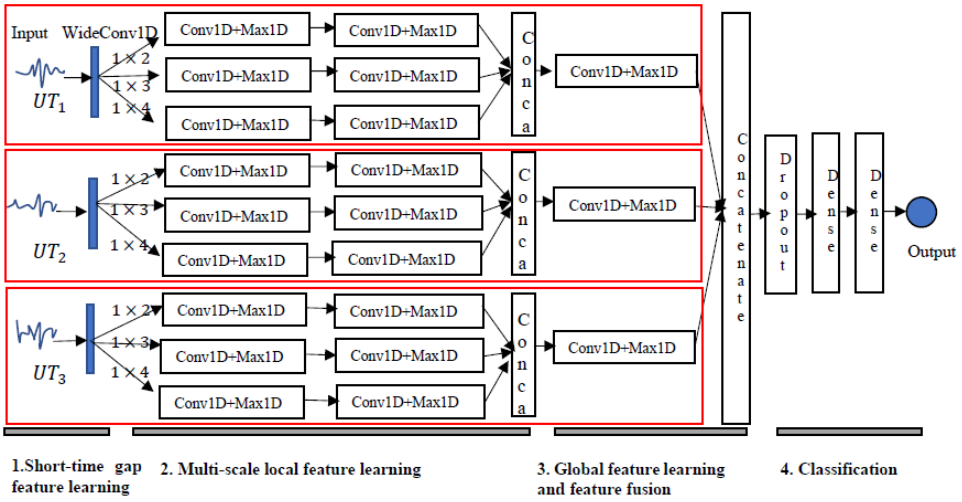


**Figure 1:** The architecture of the proposed MSFFCNN for TSC problem, which consists of four parts: short-time gap feature learning, multi-scale local feature learning, global feature learning, feature fusion, and classification, respectively. The convolution layer is donated as Conv1D, and the max-pooling layer is donated as Max1D. Additionally, 1D means convolutional, and max pooling operations are utilized to process one dimensional (1-D) tensor. The term "Conca" means concatenating operation. This architecture is given in making three-source time series ($\mathbf{UT_1}, \mathbf{UT_2}, \mathbf{UT_3}$) classification using three-scale convolution technology, which is donated as $\mathbf{1 \times 2}$, $\mathbf{1 \times 3}$, and $\mathbf{1 \times 4}$. For the UTSC

problem, only by using one part of the architecture of MSFCNN as marked in the red box for learning multi-scale and global feature representation

*3.2.1 Short-time gap feature learning*

Similar to the other CNN-based classification problems, the input shape of MSFFCNN requires 1-D tensor. Therefore, 1-D time series need to be transformed into tensor previously by using reshape operation as shown in Eq. (6) for UTSC and Eq. (7) for MTSC. Where the inputs of time series are $UT$ and $MT$, $MT$ contains some $UT$, as described in Section 2. Especially, $m$ is three in Fig. 1. For instance, by using transformation function, we could transform one $UT$ with length 100 to a tensor with the shape of $[1,100]$.

$$Tensor_{ut} = Reshape(UT) \tag{6}$$

$$Tensor_{mt} = Reshape(MT), MT = \{UT_1, UT_2, \dots, UT_m\} \tag{7}$$

After the transformation, we applied wide convolution technology to capture the features which have the most relationship in short time gaps, and we named it as short-time gap features. The wide convolution technology is implemented by one convolution layer with various big-size filters. Especially, we adopted filter size in this paper is 64, the obtained short-time feature $Feature_{stg}$ for each univariate time series could be written as Eq. (8). For MTSC, the obtained features in this sub-step are multiple $Feature_{stg}$.

$$Feature_{stg} = Conv1D(Tensor_{ut}) \tag{8}$$

*3.2.2 Multi-scale local feature learning*

The preceding feature expressions we obtained are processed by multi-scale convolution technology to extract the rich local feature representations. We defined three-scale convolution operations in Fig. 1, which are implemented through three convolution layers with different filter sizes, and each convolution layer contains various filters. We defined that three-scale convolution operations are $1 \times 2$, $1 \times 3$, and $1 \times 4$, respectively. Moreover, we adopted a max-pooling operation to decrease the dimension of features and speed up the convergence. And we call the component of combined Conv1D and Max1D as $Block$. As shown in Fig. 1, two $Block$ components existed at each scale of MSFFCNN. We formalized this process for UTSC, as shown in Eqs. (9)-(11). The filter numbers of convolution layers in $Block$ are 16 and 32, and the pooling size is 2. After getting multi-scale local feature representation, we concatenate them on the x-axis for the next process. Therefore, extracted features keep their local characteristic simultaneously have connected with different weights; it could be written as Eq. (12). For the three-source time series, three concatenated features in this sub-step are symbolized as $Feature_{conc1}$, $Feature_{conc2}$, and $Feature_{conc3}$.

$$Feature_{ms1} = Block(Block(Feature_{stg})) \tag{9}$$

$$Feature_{ms2} = Block(Block(Feature_{stg})) \tag{10}$$

$$Feature_{ms2} = Block(Block(Feature_{stg})) \tag{11}$$

$$Feature_{conc} = Concatenate(Feature_{ms1}, Feature_{ms2}, Feature_{ms3}) \tag{12}$$

*3.2.3 Global feature learning and feature fusion*

The precious sub-step already extracted multi-local rich feature representations. We utilized global convolution and max-pooling layers to capture global representations. We named this sub-step as $Static\_block$, in which the filter size is 4, and the filter number is 64. The extracted global features for UTSC are defined as Eq. (13). For MTSC, we need to concatenate multiple time series together on the $x$-axis. Then concatenated representations are processed by $Static\_block$, as shown in Eq. (14).

$$Feature_{global} = Static\_block(Feature_{conc}) \tag{13}$$

$$Feature_{mglobal} = Static\_block(Concatenate(Feature_{conc1}, Feature_{conc2}, Feature_{conc3})) \tag{14}$$

*3.2.4 Classification*

The aforementioned steps capture rich multi-scale and global feature representations; the above features are fed into two fully connected layers to extract deeper and more abstract representations. Besides, we employed a dropout layer to overcome overfitting and convergent the networks, and we set the rate of dropout layer is 0.5, it will select half of the neural nodes in the networks randomly to die when training the model. After full connection, the features could be more abstract and representative and are fed into the output layer. Output nodes depend on applications, that is how many statuses of time series have. The $softmax$ function [Liu, Wen, Yu et al. (2017)] is employed to generate the probability of each class for time series; the class corresponding to the maximum probability is the predicted label of time series. The proposed MSFFCNN only needs the raw time series for classification, as described in Eq. (15) for UTSC and Eq. (16) for MTSC. Where $MSFFCNN()$ is the trained deep model.

$$UTSC = MSFFCNN(UT) \tag{15}$$

$$MTSC = MSFFCNN(MT) = MSFFCNN(\{UT_1, UT_2, \ldots, UT_m\}) \tag{16}$$

The more configurations of our proposed MSFFCNN are given as follows: The activation function we applied is ReLU, we defined cross-entropy as loss function to update the networks, and "Adam" [Kingma and Ba (2014)] is adopted as an optimizer to tuning the loss of MSFFCNN.

## 4 Experiment verification

The platform we used in this study has an operating system of ubuntu 16.0.4 with memory 23.4 GB, Intel (R) i7-700 CPU, and processing speed 3.6 GHz.

### *4.1 UTSC verification*

*4.1.1 Data introduction*

We adopted ten data sets from UCR [Chen, Keogh, Hu et al. (2016)] for UTSC verification, which consists of binary classification and multiple classification problems, the more detailed description of UCR data sets as shown in Tab. 2. As we can see from Tab. 2, six data sets belong to binary classification problems, and four data sets are multiple classifications problems. The training case is used to train the model, and the testing case is for verification. The biggest challenging thing is that the data used for

training is much less. Additionally, different from Cui's et al. [Cui, Chen and Chen (2016) ], Guennec's et al. [Guennec, Malinowski and Tavenard (2016)], and Jiang's et al. methods [Jiang, He, Yan et al. (2018)], we only use raw time series without any data augmentation technologies to train, verify the proposed deep model. As we mentioned in 3.2.1, we set the hyperparameter of wide convolution operation filter size as 64 in MSFFCNN, the reason for that is the shortest length of time series is 65 in SonyAIBRobot2.

**Table 2:** The selected UCR univariate time series archives

| Data | Training Case | Testing case | Length | Classes |
|---|---|---|---|---|
| Wafer | 1000 | 6164 | 152 | 2 |
| TwoPatterns | 1000 | 4000 | 128 | 4 |
| SyntheticControl | 300 | 300 | 60 | 6 |
| Lightning2 | 60 | 61 | 637 | 2 |
| Powercons | 180 | 180 | 144 | 2 |
| UMD | 36 | 144 | 150 | 3 |
| Trace | 100 | 100 | 275 | 4 |
| FordA | 3601 | 1320 | 500 | 2 |
| FordB | 3636 | 810 | 500 | 2 |
| SonyAIBRobot2 | 27 | 953 | 65 | 2 |

*4.1.2 Comparative results and analysis*

To verify the effectiveness and priority of our proposed MSFFCNN for UTSC problem, we compare it with other excellent CNN-based methods: MCNN [Cui, Chen and Chen (2016)], FCNN, MLP, ResNet [Wang, Yan and Oates (2017)], Classical CNN [Zhao, Lu, Chen et al. (2017)] and MSCNN [Jiang, He, Yan et al. (2018)]. MSCNN has been proven that it is more potent than MCNN, and we implemented MSCNN (2) due to it obtained the best performance for wind turbine gearbox diagnosis. The structure of MSCNN (2) as given in Tab. 3. We also give the structure of classic CNN; the hyperparameter of classic CNN we set is the best in the author's paper. For FCNN, MLP, and ResNet, we adopted the authors' code to run. The term "raw" denotes original time series, and "mean (2)" means we adopted an overlapped method to generate the mean value of time series with stride 2. The format of convolutional operation is $Conv1D(filter\_numbers, strides)$; the default $strides$ is 1. The format of max-pooling operation is $Max1D\ (pooling\_size)$, and "AveragePool" means average pooling operation. Term of "classes" is the number of time series statues.

We adopted accuracy as the evaluation metric, and we run our proposed methods at 10 times to overcome the impact of randomness, the result is averaged accuracy on each data. The other methods' results are the best they reported could be found from Tab. 4. "Win" means the solver wined times on all those data sets, "Average Accuracy" means the averaged accuracy on those ten data sets, and we adopted standard deviation to evaluate the stability of each method. It is worthy to notice that we did not give all results of MCNN, because the author did not give all configuration information of networks, we adopted they reported values.

The findings indicate that our proposed MSFFCNN wined six best ranks and got the highest averaged accuracy on ten data sets. Even though ResNet wined the same best ranks to MSFFCNN, the averaged accuracy is only 0.9257, which is much lower than 0.9803 of the proposed method. Moreover, the standard deviation shows our proposed MSFFCNN outperforms others except for MSCNN, it is a little worse than MSCNN by comparing their standard deviation. However, MSFFCNN does not need any preprocessing operations before modelling. On the contrary, the MSCNN needs to calculate the mean values of each time series at different levels, which costs too much time and computing-resources. As a summary, our proposed MSFFCNN could predict the label of univariate time series accurately and stably without any preprocessing operations by directly inputting the original time series.

To quantify the difference between the proposed method and other leading methods listed in Tab. 4, we compute the $p$-value of the $t$-test, as shown in Tab. 5. The results show that all those methods are the same distributions at a confidence level of 95%. The reason of that is we selected methods already act as a leading role for UTSC. Moreover, the proposed method and MSCNN could be divided into the first group for UTSC because the $p$-value of them is near to 1 and average accuracy of them is too much similar around 0.98; Zhao's CCNN and ResNet could be divided into the second group because their $p$-values are higher than 0.2. MLP and FCN could be divided into the last group.

**Table 3:** The configurations of comparative CNN-based methods for UTSC

| Method | Description |
|---|---|
| MSCNN [Jiang, He, Yan et al. (2018)] | Three inputs: $Input(raw, mean(2), mean(3))$<br>Sub1:<br>Raw-Conv1D(16)-Max1D(2)-Conv1D(32)-Max1D(2)-Conv1D(64)-Max1D(2)<br>Sub2:<br>Mean(2)-Conv1D(16)-Max1D(2)-Conv1D(32)-Max1D(2)-Conv1D(64)-Max1D(2)<br>Sub3:<br>Mean(3)-Conv1D(16)-Max1D(2)-Conv1D(32)-Max1D(2)-Conv1D(64)-Max1D(2)<br>Output: Concatenate(Sub1, Sub2,Sub3)-Dense(100)-Dense(classes)<br>Activation function is "ReLU" and optimizer is "Adam", loss function is "cross-entropy". |
| Classical CNN [Zhao, Lu, Chen et al. (2017)] | Raw-Conv1D(7,6)-AveragePool(3)-Conv1D(7,12)-AveragePool(3)-Dense(classes)<br>Activation function is "ReLU" and optimizer is "Adam", Loos function is Mean Square Error. |

**Table 4:** Testing accuracy and averaged accuracy on 10 UCR sensor-related data sets

| | Proposed | CCNN | MCNN | MLP | FCN | ResNet | MSCNN |
|---|---|---|---|---|---|---|---|
| Wafer | **1.0** | 0.9970 | 0.9980 | 0.9960 | 0.9970 | 0.9970 | **1.0** |
| TwoPatterns | 0.9996 | 0.9650 | 0.9980 | 0.8860 | 0.8970 | **1.0** | 0.9979 |
| SyntheticControl | 0.9886 | 0.9730 | 0.9970 | 0.9500 | 0.9900 | **1.0** | 0.9950 |
| Lightning2 | **1.0** | **1.0** | 0.8360 | 0.7210 | 0.8030 | 0.7540 | **1.0** |
| Powercons | **1.0** | **1.0** | - | 0.9722 | 0.5056 | 0.8778 | 0.9990 |
| UMD | 0.9700 | 0.9653 | - | 0.9722 | 0.5833 | 0.5694 | **0.9800** |
| Trace | **1.0** | 0.6400 | **1.0** | 0.9200 | **1.0** | **1.0** | 0.9700 |
| FordA | **1.0** | **1.0** | - | **1.0** | **1.0** | **1.0** | **1.0** |
| FordB | **1.0** | **1.0** | - | **1.0** | **1.0** | **1.0** | **1.0** |
| SonyAIBRobot2 | 0.8448 | 0.8279 | 0.9300 | 0.7270 | 0.9680 | **0.9850** | 0.8400 |
| Win (Best ranks) | 6 | 4 | - | 2 | 3 | **6** | 5 |
| Averaged Accuracy (AVG) | **0.9803±0.0461** | 0.9368 **±0.1107** | - | 0.9138 **±0.1013** | 0.8744 **±0.1763** | 0.9257 **±0.1350** | 0.9801 **±0.0453** |

**Table 5:** The $\boldsymbol{p}$-value of comparison results using $\boldsymbol{t}$-test

| | CCNN | MLP | FCN | ResNet | MSCNN |
|---|---|---|---|---|---|
| Proposed | 0.281 | 0.093 | 0.098 | 0.222 | **0.925** |

*4.1.3 Feature extraction capacity validation*

To explore and validate MSFFCNN's features extraction capacity. Firstly, we have visualized the activated output of the convolutional layer in MSFFCNN using t-SNE technology for reducing the dimension of extracted feature maps, as shown in Fig. 2. The data we adopted is SyntheticControl. From Fig. 2, we can see feature representations in (a) are mixed, difficult to identify, inseparable, and none-linear. After one-dimensional convolution with a large size filter, the feature expressions are becoming separable, which could be found from (b). After the multi-feature extraction process, learned feature repressions are discriminable, and independent as shown in (c), which satisfies the principle of classification: the maximum differences in an interclass, minimum variation in the same class. Additionally, the global feature learning step ensures more abstract and vibrant feature expressions are extracted from the preceding layer. The comprehensive feature representations, as shown in (d). The results indicate our proposed method could capture most of the useful representations for accurately predicting time series label with a cascading structure of CNN.

Secondly, we have analyzed the inside feature maps to confirm the productive feature extraction capacity of our proposed method again. The visualization results using one sample from the SyntheticControl, as shown in Fig. 3. We can see the original time series oscillate with time stamp, and the range is from -2 to 2, as shown in Fig. 3(a). The short time feature map is obtained by one wide convolution operation, as shown in Fig. 3(b), it decreases from 2.0 to 1.75, the reason of that is the original time series consists of some noisy data points. Additionally, the reason that only positive values occurred in the feature map is converting the values of time series into RGB representations. Most of the parts in

Fig. 3(b) are blue, which donates detailed information. The range of multi-scale feature maps increased from 1.75 of the short-time gap feature into 2.5, which can be seen in Fig. 3(c). It is easier to identify each time series through a multi-scale feature map because the values of the feature map increase a lot. Through the global feature learning process, the feature map is clear and identifiable, because it increased some lager values and decreased some white points with a lower value, which could be found from Fig. 3(d).

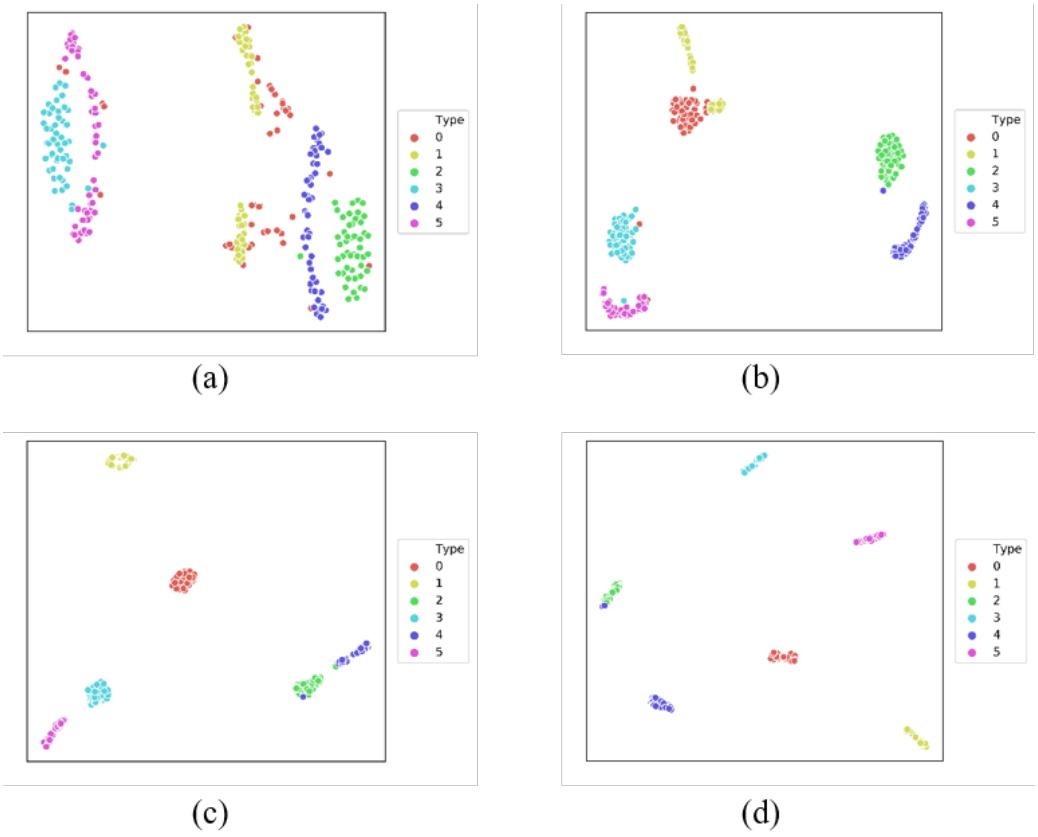


**Figure 2:** Feature visualization via t-SNE technology reduced from MSFCNNs using SyntheticControl data. (a) raw signals. (b) one-dimensional convolution results after a short time gap convolutional operation. (c) the result after multi-local convolution operations. (d) combined multi-local and global feature repressions

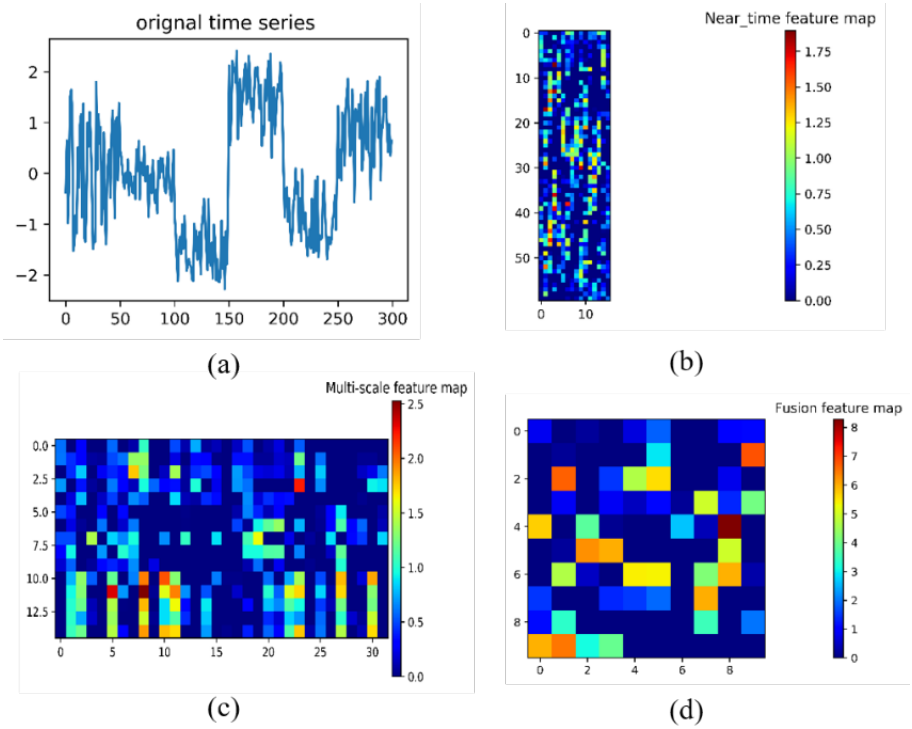


**Figure 3:** Feature map visualization. (a) one raw time series. (b) extracted short-time gap feature map using wide convolution technology. (c) learned multi-scale feature map. (d) combined multi-local feature and global feature as one comprehension fusion feature map, we reshape the original feature map with 100 into $10 \times 10$ to better display

*4.1.4 The influence of different scale*

The above analysis is based on three-scale MSFFCNN. Different scales may influence the performance of classification. We have compared two-scale MSFFCNN (MSFFCNN (2)) and four-scale MSFFCNN (MSFFCNN (4)) to explore the influence of scale-level. The configuration of MSFFCNN (2) and MSFFCNN (4) are the same as MSFFCNN (3) we give in Fig. 1 expect for the scale level. We designed MSFFCNN (2) with $1 \times 2$ and $1 \times 3$ convolution operations, and MSFFCNN (4) with $1 \times 2$, $1 \times 3$, $1 \times 4$, and $1 \times 5$ convolution operations. The results as shown in Tab. 6. The findings show the accuracy increased with increasing of the scale level, and it is more stable when we apply more scales. We did the significance test using the $t$-test, which indicates that there is no significant difference between these three methods. The pairwise $p$-value of MSFFCNN (2) and MSFFCNN (3) is 0.962, MSFFCNN (4) and MSFFCNN (3) is 0.920, respectively. Another creditable phenomenon is that it is more stable with the increasing of scale level, which can be seen from the standard deviation in Tab. 6. It is worthy to notice that as the scale level increases, it needs more time to train and test. Therefore, we adopted MSFFCNN (3) for UTSC to balance the time resource while simultaneously keeping the high performance.

**Table 6:** Comparison results of three different models on 10 UCR sensor-related data sets

| | MSFFCNN (2) | MSFFCNN (3) | MSFFCNN (4) |
|---|---|---|---|
| Wafer | **1.0** | **1.0** | 1.0 |
| TwoPatterns | 0.9998 | 0.9996 | 1.0 |
| SyntheticControl | 0.9900 | 0.9886 | 0.9900 |
| Lightning2 | **1.0** | **1.0** | 1.0 |
| Powercons | **1.0** | **1.0** | 1.0 |
| UMD | 0.9722 | 0.9700 | 0.9861 |
| Trace | **0.9900** | **1.0** | **1.0** |
| FordA | **1.0** | **1.0** | 1.0 |
| FordB | **1.0** | **1.0** | 1.0 |
| SonyAIBRobot2 | 0.8405 | 0.8448 | 0.8488 |
| AVG | **0.9793±0.0470** | **0.9803±0.0461** | 0.9824±**0.0448** |

### *4.2 MTSC verification*

The above analysis has confirmed the effectiveness and priority of MSFFCNN for UTSC. We also designed a concise experiment to verify the progressiveness of MSFFCNN for the MTSC problem as follows.

#### *4.2.1 Data introduction*

Two data sets we adopted to validate the effectiveness of MSFFCNN for the MTSC problem, they are WISDM v1-split and WISDM v.2, which is same to Yazdanbakhsh's et al. paper [Yazdanbakhsh and Stick (2019)]. WISDM v1-spilt consists of accelerometer data collected from 36 users regarding their daily six activities, including walking, jogging, upstairs, downstairs, sitting, and standing. WISDM v.2 consists of accelerometer data collected from 56 users while walking, jogging, stairs, sitting, standing, and lying down. 41279 samples are generated as the training part and 13162 samples as a testing part in WISDM v1-split, respectively. For WISDM v.2, giving 10396 training samples and 4456 testing samples. The detailed description of data sets and generation method could be found from Yazdanbakhsh's et al. paper [Yazdanbakhsh and Stick (2019)].

#### *4.2.2 Comparative analysis*

The evaluation metric for MTSC is F-1 scores of each label, and we compared our method to dilated CNN [Yazdanbakhsh and Stick (2019)], MCDCNN(2) [Zheng, Liu, Chen et al. (2014)], one feature-based method [Ravi, Wong, Lo et al. (2017)] named Ravelet, and classic CNN. We implement multiple classic CNN for MTSC based on Zhao's et al. method [Zhao, Lu, Chen et al. (2017)]. The comparison results are summarized in Tabs. 7 and 8.

The findings from Tab. 7 indicate our proposed method outperforms other CNN-based methods, and a little lower than the feature-based method by comparing the averaged accuracy. We did a $t$-test to quantify this difference between the proposed method and Ravelet's method. The result indicates there is no significant difference due to the $p$-value

(0.624) is much higher than 0.05. Moreover, our proposed method does not need any feature selection operation. By contraries, Ravelet's method needs. All those methods perform well on WISDM v1-spilt data, whose averaged accuracies are higher than 90.0%.

The findings from Tab. 8 indicate our proposed method outperforms others, its averaged accuracy up to 92.8%. It has many improvements compared to other state-of-the-art methods with metric of averaged accuracy. It improved 3.5% compared to classic CNN [Zhao, Lu, Chen et al. (2017)], 30% to MCDCNN (2) [Zheng, Liu, Chen et al. (2014)], 4% to dilated CNN [Yazdanbakhsh and Stick (2019)], respectively. Moreover, it shows our method could accurately predict all kinds of labels over accuracy of 93% except for the activity of "Sitting." MCDCNN (2) almost cannot predict the labels of "sitting" and "standing." In summary, our proposed method can accurately handle the MTSC problem without any preprocessing and feature selection operations.

**Table 7:** The result of WISDM v1-split based on F1 score

| | Proposed | Classic CNN | MCDCNN(2) | Dilated CNN | Ravelet |
|---|---|---|---|---|---|
| Walking | 98.0% | 98.8% | 98.4% | 97.4% | 99.3% |
| Jogging | 99.5% | 98.3% | 99.3% | 98.3% | 99.5% |
| Upstairs | 93.3% | 89.5% | 93.5% | 86.4% | 95.3% |
| Downstairs | 90.0% | 91.9% | 81.3% | 80.5% | 95.1% |
| Sitting | 99.2% | 98.9% | 96.9% | 98.0% | 98.2% |
| Standing | 99.5% | 97.8% | 98.4% | 94.9% | 97.6% |
| Average | 96.4% | 95.9% | 94.63% | 92.58% | 97.5% |

**Table 8:** The result of WISDM v2 based on F1 score

| | Proposed | Classic CNN | MCDCNN (2) | Dilated CNN | Ravelet |
|---|---|---|---|---|---|
| Walking | 97.9% | 95.9% | 95.6% | 96.6% | 97.2% |
| Jogging | 98.9% | 94.0% | 90.6% | 96.9% | 97.9% |
| Upstairs | 93.5% | 90.7% | 96.0% | 63.1% | 79.3% |
| Downstairs | 94.7% | 87.5% | 82.9% | 91.2% | 88.2% |
| Sitting | 75.2% | 53.3% | 2.2% | 87.2% | 82.1% |
| Standing | 96.3% | 84.6% | 4.7% | 90.7% | 87.2% |
| Average | 92.8% | 89.3% | 62.0% | 87.6% | 88.7% |

## 5 Discussion

We have proposed a novel deep model named MSFFCNN to extract productive and robust feature representations of raw sensor-related time series for predicting their labels automatically and accurately. Predicting the label of time series could be expressed as one TSC problem including UTSC and MTSC. The difficulty of UTSC is that training samples is much less than testing samples, as shown in Tab. 2, which requires the model could extract rich and robust feature representations from rare training samples. Besides, some CNN-based method still performs worse than feature-based method. Therefore, we designed one novel CNN-based deep model to capture multi-scale and global fusion

features to overcome the above issues, as shown in Fig. 1.

We have compared our proposed method with other excellent deep learning-based models on ten UCR sensor-related data sets, as shown in Tab. 4, it indicates our proposed MSFFCNN is most competitive by using averaged accuracy, and also it is stable by comparing it to others in terms of standard deviation. To quantify this difference, we have calculated the $p$-value of the $t$-test, the results confirmed our proposed model belongs to the first class for UTSC, as shown in Tab. 5.

As can be seen from Fig. 2, we have analyzed the feature learning capacity of the proposed MSFFCNN using t-SNE technology. It explained that wide convolution technology could extract more robust feature maps through raw time series, multi-scale feature learning sub-process could learn features at the multi-scale level. Also, fusion features are robust. The feature extracting capacity of our proposed model has been proven. Also, we have analyzed the interior feature to confirm the feature extraction capacity again, as shown in Fig. 3.

We designed three models at different scale levels to analyze the impact of scale level, as shown in Tab. 6. The findings show the accuracy increases with the scale level, and it is more stable when we apply more scales. And we have proven that there is no significant difference when we utilize our model at different scale levels by computing the $p$-value of the $t$-test. Therefore, our proposed structure is stable and has a good generalization ability.

As shown in Tabs. 7 and 8, we have designed, trained, and evaluated our proposed MSFFCNN for MTSC. The results indicate that our proposed model has state-of-the-art performance for MTSC without any preprocessing and handcrafted feature selection operations.

## 6 Conclusion

In conclusion, we have proposed a new, accurate, and stable approach for time series classification based on CNN. After a set of the feature extraction process in the proposed MSFFCNN, we could obtain multi-scale, global, and robust fusion feature representations. Our experiments show that our proposed method could predict the label of both univariate and multivariate time series accurately and automatically without any handcrafted feature engineering by using less training data set. In addition, the proposed framework is very stable, which is not sensitive to the scale levels.

We already tested the effectiveness and priority of the proposed method for time series classification problem. In the future, we will utilize the proposed MSFFCNN to process the Job Scheduling Problem (JSP), in which we consider JSP as one classification problem, MSFFCNN would deal with JSP as well.

**Funding Statement:** This work was supported by the Technology Innovation Program (20004205, The development of smart collaboration manufacturing innovation service platform in textile industry by producer-buyer B2B connection funded By the Ministry of Trade, Industry & Energy (MOTIE, Korea)).

**Conflicts of Interest:** The authors declare no conflicts of interest.

**References**

**Ahn, J.; Lee, J.** (2018): Clustering algorithm for time series with similar shapes. *KSII Transactions on Internet and Information Systems*, vol. 12, no. 7, pp. 3112-3127.

**Arathi, M.; Govardhan, A.** (2015): Effect of mahalanobis distance on time series classification using shapelets. *Proceedings of the 49th Annual Convention of the Computer Society of India*, vol. 2015, pp. 525-534.

**Arif, S.; Wang, J.; Fei, Z.; Hussain, F.** (2019): Video representation via fusion of static and motion features applied to human activity recognition. *KSII Transactions on Internet and Information Systems*, vol. 13, no. 7, pp. 3599-3619.

**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*, vol. 27, no. 9, pp. 2522-2535.

**Batista, G. E. A. P. A.; Wang, X.; Keogh, E. J.** (2011): A complexity-invariant distance measure for time series. *Proceedings of the 11th SIAM International Conference on Data Mining*, vol. 2011, pp. 699-710.

**Cao, D.; Tian, Y.; Bai, D.** (2015): Time series clustering method based on principal component analysis. *5th International Conference on Information Engineering for Mechanics and Materials*, pp. 888-895. https://doi.org/10.2991/icimm-15.2015.163.

**Chen, Y. T.; Xu, W. H.; Zuo, J. W.; Yang, K.** (2019): The fire recognition algorithm using dynamic feature fusion and IV-SVM classifier. *Cluster Computing*, vol. 22, no. 3, pp. 7665-7675.

**Chen, Y.; Keogh, E.; Hu, B; Begum, N.; Bagnall, A. et al.** (2016): The ucr time series classification archive. https://www.cs.ucr.edu/~eamonn/time_series_data/.

**Chen, Z. Q.; Li, C.; Sanchez, R. V.** (2015): Gearbox fault identification and classification with convolutional neural networks. *Shock and Vibration*, vol. 2015, no. 2, pp. 1-10.

**Chotirat, A. R.; Eamonn, K.** (2005): Three myths about dynamic time warping data mining. *Proceedings of SIAM International Conference on Data Mining*, vol. 2005, pp. 506-510.

**Cui, Z.; Chen, W.; Chen, Y.** (2016): Multi-scale convolutional neural networks for time series classification. [Online], Available: https://arxiv.org/abs/1603.06995.

**Guennec, A. L.; Malinowski, S.; Tavenard, R.** (2016): Data augmentation for time series classification using convolutional neural networks. *Proceedings of the ECML/PKDD Workshop on Advanced Analytics and Learning on Temporal Data*, Riva Del Garda, Italy.

**Hendrik, D. B.** (2008): Fourier transform and related integral transforms in superspace. *Journal of Mathematical Analysis and Applications,* vol. 345, no. 1, pp. 147-164.

**Hills, J.; Lines, J.; Baranauskas, E.; Mapp, J.; Bagnall, A.** (2014): Classification of time series by shapelet transformation. *Data Mining and Knowledge Discovery*, vol. 28, no. 4 pp. 851-881.

**Jiang, G. Q.; He, H. B.; Yan, J.; Xie, P.** (2018): Multiscale convolutional neural networks for fault diagnosis of wind turbine gearbox. *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3196-3207.

**Kingma, D.; Ba, J.** (2017): Adam: a method for stochastic optimization. https://arxiv.org/abs/1412.6980.

**Kwapisz, J. R.; Weiss, G. M.; Moore, S. A.** (2011): Activity recognition using cell phone accelerometers. *ACM SigKDD Explorations Newsletter*, vol. 12, no. 2, pp. 74-82.

**Lecun, Y.; Bengio, Y.; Hinton, G.** (2015): Deep learning. *Nature*, vol. 521, pp. 436-444.

**Lei, Y.; Wu, Z.** (2020): Time series classification based on statistical features. *EURASIP Journal on Wireless Communications and Networking*, vol. 2020, no. 1.

**Li, H. L.** (2016): Accurate and efficient classification based on common principal components analysis for multivariate time series. *Neurocomputing*, vol. 171, pp. 744-753.

**Lines, J.; Bagnall, A.** (2015): Time series classification with ensembles of elastic distance measures. *Data Mining and Knowledge Discovery*, vol. 29, no. 3, pp. 565-592.

**Liu, C.; Hsaio, W.; Tu, Y.** (2019): Time Series classification with multivariate convolutional neural network. *IEEE Transactions on Industrial Electronics*, vol. 66, no. 6, pp. 4788-4797.

**Liu, J.; Yang, Y. H.; Lv, S. Q.; Wang, J.; Chen, H.** (2019): Attention-based BiGRU-CNN for chinese question classification. *Journal of Ambient Intelligence and Humanized Computing*. https://doi.org/10.1007/s12652-019-01344-9.

**Liu, W. Y.; Wen, Y. D.; Yu, Z. D.; Yang, M.** (2017): Large-margin softmax loss for convolutional neural networks. https://arxiv.org/abs/1612.02295.

**Lockhart, J. W.; Weiss, G. M.; Xue, J. C.; Gallagher, S. T.; Grosner, A. B. et al.** (2011): Design considerations for the wisdm smart phone-based sensor mining architecture. *Proceedings of the Fifth International Workshop on Knowledge Discovery from Sensor Data*, vol. 2011, pp. 25-33.

**Peng, H. K.; Marculescu, R.** (2015): Multi-scale compositionality: identifying the compositional structures of social dynamics using deep learning. *PLoS One*, vol. 10, no. 4, e0118309.

**Ravi, D.; Wong, C.; Lo, B.; Yang, G. Z.** (2017): A deep learning approach to on-node sensor data analytics for mobile or wearable devices. *IEEE Journal of Biomedical and Health Informatics*, vol. 21, no. 1, pp. 56-64.

**Wang, Z.; Yan, W.; Oates, T.** (2017): Time series classification from scratch with deep neural networks: a strong baseline. *International Joint Conference on Neural Networks*, vol. 2017, pp. 1578-1585.

**Xing, Z.; Pei, J.; Keogh, E.** (2010): A brief survey on sequence classification. *ACM SIGKDD Explorations Newsletter*, vol. 12, no. 1, pp. 40-48.

**Yazdanbakhsh, O.; Dick, S.** (2019): Multivariate time series classification using dilated convolutional neural network. https://arxiv.org/abs/1905.01697.

**Zhang, H.; Ho, T. B.; Lin, M. S.; Liang, X.** (2006): Feature extraction for time series classification using discriminating wavelet coefficients. *Advances in Neural Networks-ISNN, Lecture Notes in Computer Science*, vol. 397, pp. 1394-1399.

**Zhao, B.; Lu, H.; Chen, S.; Liu, J.; Wu, D.** (2017): Convolutional neural networks for time series classification. *Journal of Systems Engineering and Electronics*, vol. 28, no. 1, pp. 162-169.

**Zheng, Y.; Liu, Q.; Chen, E.; Ge, Y.; Zhao, J. L.** (2014): Time series classification using multi-channels deep convolutional neural networks. *Web-Age Information Management, Lecture Notes in Computer Science*, vol. 8485, pp. 298-310.