Knowledge-Based Systems 243 (2022) 108459

Contents lists available at ScienceDirect

Knowledge-Based Systems

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

Adaptive feature fusion for time series classification

Tian Wang [a], Zhaoying Liu [a], Ting Zhang [a,*], Syed Fawad Hussain [b], Muhammad Waqas [c,d], Yujian Li [a,e]

[a] *Engineering Research Center of Intelligent Perception and Autonomous Control, Faculty of Information Technology, Beijing University of Technology, 100124 Beijing, China*
[b] *Faculty of Computer Science and Engineering, Ghulam Ishaq Khan Institute of Engineering Sciences and Technology, Topi 23460, Pakistan*
[c] *College of Information Technology, Computer Engineering Department, University of Bahrain, 32038, Bahrain*
[d] *School of Engineering, Edith Cowan University, Perth WA, 6027, Australia*
[e] *School of Artificial Intelligence, Guilin University of Electronic Technology, Guilin 541004, China*

A R T I C L E I N F O

Article history:
Received 21 October 2021
Received in revised form 9 February 2022
Accepted 16 February 2022
Available online 23 February 2022

Keywords:
Time series classification
Multi-scale temporal features
Distance features
Distance prototype network
Adaptive feature fusion

A B S T R A C T

Time series classification is one of the most critical and challenging problems in data mining, which exists widely in various fields and has essential research significance. However, to improve the accuracy of time series classification is still a challenging task. In this paper, we propose an Adaptive Feature Fusion Network (AFFNet) to enhance the accuracy of time series classification. The network can adaptively fuse multi-scale temporal features and distance features of time series for classification. Specifically, the main work of this paper includes three aspects: firstly, we propose a multi-scale dynamic convolutional network to extract multi-scale temporal features of time series. Thus, it retains the high efficiency of dynamic convolution and can extract multi-scale data features. Secondly, we present a distance prototype network to extract the distance features of time series. This network obtains the distance features by calculating the distance between the prototype and embedding. Finally, we construct an adaptive feature fusion module to effectively fuse multi-scale temporal and distance features, solving the problem that two features with different semantics cannot be effectively fused. Experimental results on a large number of UCR datasets indicate that our AFFNet achieves higher accuracies than state-of-the-art models on most datasets, as well as on the WISDM, HAR and Opportunity datasets, demonstrating its effectiveness.

1. Introduction

A time series is considered a set of values arranged in chronological order. In today's big data era, time-series data is ubiquitous in life, such as financial data [1], medical data [2], and industrial data [3]. Analysis of these data can assist people in making work and life decisions. Time series classification (TSC) refers to identifying a sequence's category by analyzing features that help it distinguish itself from other sequences. Unfortunately, time series are often characterized by high data dimensionality, considerable noise and unknown time-dependent length, which increase the TSC difficulty [4].

Extensive research has been conducted on TSC, where various methods have been proposed, and many achievements have been made. With earlier methods, euclidean distance (ED) [5] or dynamic time warping (DTW) [5] is often used to calculate the distance between time series first, followed by the nearest neighbor-based classification. Later, scholars attempted to extract various features of time series, such as statistical features [6], shapelets features [7], frequency features [8]. Subsequently, scholars also put forward ensemble models, including HIVE-COTE [9] and the TS-Chief [10].

Although the methods mentioned above somewhat improved the accuracy of TSC, certain costs still need to be paid. For instance, the distance-based nearest neighbor classification requires the computation of the distance between time series, which is tremendously burdensome. In addition, the feature-based methods require manual extraction of time series features and are often cumbersome processes. Moreover, the ensemble models generally require multiple classifiers, which result in complicated modeling, high computational complexity and long training time.

With the success of deep learning in various fields [11–14], researchers have begun to address the TSC problems by utilizing neural networks. Among numerous neural network models, multilayer perceptron (MLP), convolutional neural network (CNN)

* Corresponding author.
E-mail addresses: wt9636@163.com (T. Wang), zhaoying.liu@bjut.edu.cn (Z. Liu), zhangting@bjut.edu.cn (T. Zhang), fawadhussain@giki.edu.pk (S.F. Hussain), engr.waqas2079@gmail.com (M. Waqas), liyujian@bjut.edu.cn (Y. Li).

https://doi.org/10.1016/j.knosys.2022.108459
0950-7051/© 2022 Elsevier B.V. All rights reserved.

and echo state network (ESN) are the three extensively used models in the TSC tasks [15]. CNN's gain relatively wider application due to their preferable performance. A number of CNN models have been proposed for TSC tasks, such as multi-channel CNN (MC-CNN) [16] and multi-scale convolutional neural network (MCNN) [17]. The problem is that most of the existing CNN models extract convolutional features only. At the same time, the time series data has a small size and does not contain rich information like image data, whose characteristics can hardly be obtained fully through mere reliance on the convolutional features.

For extraction of more features, model like Omni-scale 1D-CNN (OS-CNN) [18] use various sizes of convolutional kernels to extract multi-scale features, achieving specific effects. In other methods, Iwana [19] describes the time series characteristics based on the local distance information, where the local DTW distance is considered. However, this type of method only takes into account the distance features. Thus, despite resolving simple and small datasets preferably, distance information's classification ability is somewhat limited for some complex time series data.

From the above, it is clear that most of the existing deep learning-based methods use an end-to-end deep neural network to extract a type of times series features, such as convolution or distance features, which improves the accuracy of TSC to some extent. However, the information contained in a single class of features is usually limited, it can hardly describe the time series comprehensively and accurately, thereby limiting the neural network performance.

To address the preceding problem, we propose a novel end-to-end network architecture, adaptive feature fusion network (AFFNet), to further improve the performance of univariate time series classification. It sequentially extracts the multi-scale temporal and distance features of time series first. Then, it integrates these two features through an adaptive feature fusion module to obtain richer fused features. Finally, it performs the classification based on these fused features. Specifically, this proposed model comprises three parts: a multi-scale dynamic convolution network (MDCNet) used for extracting the multi-scale temporal features, a distance prototype network (DPNet) used for extracting the distance features, and an adaptive feature fusion module (AFFM) for integrating the two types of extracted features.

In summary, the major contributions of this paper include five aspects, i.e.,

1. A TSC model based on adaptive feature fusion, called AFFNet, is proposed. It sequentially extracts the multi-scale temporal features and the distance features of time series and integrates these two types of features through the AFFM. By classifying time series based on the fused features, the performance of TSC is improved.
2. A multi-scale dynamic convolution network, named MDCNet, is presented. It extracts the multi-scale temporal features of time series through multi-scale dynamic convolution and effectively processes different data to obtain multi-scale time-series features.
3. A distance prototype network, with the name of DPNet, is offered. It extracts the distance features of time series in the embedded space by optimizing the distance between the prototype and the embedded vector. By supplementing the multi-scale temporal features with distance features, the characteristics of time series data are more fully described.
4. An adaptive feature fusion module of features, whose name is AFFM, is constructed. It can effectively integrate the multi-scale temporal and distance features to obtain the fusion features with more vital expressive ability.
5. Experimental results demonstrate that AFFNet outperforms the previous state-of-the-art models on 85 UCR datasets, as well as on WISDM, HAR and Opportunity datasets.

The remainder of this paper is arranged as follows. First, Section 2 introduces the related works briefly. Then, Section 3 describes our model and its modules in detail. Next, Section 4 presents the experimental results and performance analysis. Finally, Section 5 concludes this work.

2. Related works

Our work is primarily related to three tasks: time series classification, dynamic convolution and feature fusion.

2.1. Time series classification

Existing TSC methods can be roughly classified into four categories, i.e., distance-based methods, feature-based methods, ensemble-based methods, and deep learning-based methods [20]. The distance-based methods often calculate the euclidean distance (ED) or dynamic time warping (DTW) distance between pairwise time series before performing classification via k-nearest neighbor classifiers [5]. Other distance-based approaches include using distances to obtain new feature representations of series and using distance measure to construct kernels for time series [21,22]. The feature-based methods extract the time series features through manual design first and then perform classifier-based classification. For example, Schafer [23] put forward the BOSS model for TSC. The model converts sequences into characters and uses the idea of bag-of-words to extract frequency features and then performs nearest neighbor classification. It achieves good results on the UCR datasets. Dempster et al. [24] proposed a Rocket model for TSC. Initially, the model uses many random convolutional kernels to perform convolution operations on series and then extracts the operation results as features, followed by a linear classifier to classification. This model dramatically improves the training speed while ensuring accuracy. Rakthanmanon et al. [25] proposed the fast shapelet (FS) model, which accelerates the shape search process by converting data into discrete low-dimensional representations, thereby significantly reducing the shapelets algorithm complexity. Grabocka et al. [26] developed the learned shapelets (LS) model, which uses the gradient descent algorithm to learn the most discriminative shape and avoids selecting candidate shapelets from substantial series.

The ensemble-based methods realize the classification of time series data by fusing multiple different types of classifiers. For example, the elastic ensemble (EE) [27] model integrates 11 nearest neighbor classifiers for TSC, whose performance is superior to any single classifier. The collective of transformation-based ensembles (COTE) [28] integrates 35 classifiers, where each classifier uses one of the temporal features in the time, frequency or shape domain to achieve classification, and the classified category is finalized through weighted voting. This model greatly improves the classification efficiency of conventional methods.

Owing to its automatic feature extraction and good generalization abilities, deep learning has attained excellent results in the fields of computer vision [29], and natural language processing [30]. Accordingly, scholars have researched deep learning-based TSC, proposing a variety of methods. For example, Wang et al. [31] used MLP, FCN and ResNet to classify raw data, thereby avoiding the complex data preprocessing and realizing the end-to-end learning process, which provides a baseline for the deep learning-based TSC. Kashiparekh et al. [32] captured multi-scale temporal features using various sizes of convolution kernels in the convolutional layer and trained a better model through fine-tuning. Borrowing the architecture of Inception [33], Fawaz et al. [34] proposed the InceptionTime model to capture features through convolution with multiple different kernel sizes in the same convolutional layer, which achieves high classification accuracy. Compared to the conventional ensemble methods, the training speed is greatly improved. Zhang et al. [35] proposed the TapNet, which learns attention prototypes in the embedded space and performs classification by calculating the distance between prototype and series embedding, thereby addressing the small sample size problem to some extent. Ma et al. [36] proposed the echo memory-augmented network (EMAN) model, which extracts the echo status information of series with echo state networks (ESN) [37] and enhances the expression of the information by designing an echo memory-augmented mechanism. This model resolves the long-term dependence problem with ESN. Wang et al. [20] developed a TSC network based on fused features (TSC-FF), where sequences are converted into images to extract visual features with CNN. Meanwhile, the sequence features of data are extracted using LSTM, followed by a final concatenation and classification of two features. Their model achieved good efficiency on the UCR datasets.

2.2. Dynamic convolution

The purpose of dynamic convolution is to find a balance between network performance and computational load [38], intending to improve the overall model performance by increasing a few parameters. Generally, dynamic convolution is realized by generating weights through the attention mechanism and fusing several static convolutions. Many studies have demonstrated that dynamic convolution can effectively upgrade the performance of existing models. For example, Yang et al. [39] proposed the conditionally parameterized convolutions (CondConv) to fuse multiple convolution kernels through the ROUTE function-based calculation of weights. Capable of obtaining results via a single convolution calculation, the model performance is improved while maintaining inference speed. The dynamic convolution (DynamicConv) proposed by Chen et al. [38] calculates the attention weight of convolution kernels using the squeeze-and-excitation (SE) module [40].

Meanwhile, to improve the training efficiency of convolution kernels, a smooth attention method is designed to optimize multiple convolution kernels simultaneously. From the perspective of uncorrelated noise features, Zhang et al. [41] used the coefficient prediction and dynamic generation modules to generate dynamic convolution. Experimental results demonstrate that the dynamic convolution kernels have a lower correlation and improve model performance compared to ordinary convolution. Ma et al. [42] investigated the relationship between CondConv and SENet, and attempted to unify the two into the same space to propose the WeightNet model. Experimental results reveal that their method can control the amount of model calculation while ensuring accuracy. Given the large parameter number and optimization difficulty of dynamic convolution, Li et al. [43] replaced dynamic fusion with the dynamic attention over channel groups from the matrix factorization perspective, which reduces the number of parameters required for dynamic convolution without affecting the model performance. Qian et al. [44] proposed Dynamic Multi-Scale Convolutional Network which can obtain convolution kernels with different lengths according to the input. It can extract multi-scale feature for each time series flexibly and achieved good results on UCR datasets.

2.3. Feature fusion

Feature fusion is a common technical measure in deep learning. The addition (Add) and concatenation (Concat) operations, in particular, have been applied in many network models. For example, ResNet [12] and feature pyramid network (FPN) [45] both use the "Add" operation to fuse feature maps, while Inception [33] and DenseNet [46] use the "Concat" operation to combine feature maps. In most cases, these two fusion methods are effective.

However, for two types of features that are complex and even semantically different, simple add or concat is not a good choice. To cope with various fusion scenarios to obtain better features, Dai et al. [47] proposed the attentional feature fusion, which provides a unified and universal method for feature fusion and applies to various networks such as Inception and FPN. Experimental results prove the ability of attentional feature fusion to improve the model performance. Zhou et al. [48] put forward the collaborative index embedding. With this method, indexes are built separately for two different features. By optimizing the index embedding, the two types of feature information are mutually fused to improve the feature expressiveness. This algorithm achieves excellent results in image retrieval tasks.

In this paper, We design an adaptive feature fusion module for the better fusion of different features, which integrates multi-scale temporal and distance features. The proposed module can realize the fusion of feature information through the interaction between two feature types. The module also augments the expressiveness of features during the interaction to obtain more discriminative fused features.

3. Method

In this section, we first introduce the overall structure of the Adaptive Feature Fusion Network (AFFNet). Then, we describe the Multi-scale Dynamic Convolution Network (MDCNet), Distance Prototype Network (DPNet) and Adaptive Feature Fusion Module (AFFM). Next, we give an analysis of its model complexity. Finally, it gives the loss function of the training procedure.

3.1. Network architecture

We propose an adaptive feature fusion network architecture for time series classification. First, it sequentially extracts the multi-scale temporal features and distance features of time series data. Then, it integrates these two feature types through the adaptive feature fusion module, ultimately achieving the classification of fused features. As is clear from Fig. 1, which depicts the architecture of AFFNet, the network comprises three parts: MDCNet, DPNet and AFFM. Specifically, MDCNet extracts the multi-scale temporal features of time series through the multi-scale dynamic convolution. Afterwards, DPNet maps the temporal features into the embedded space by deeming the multi-scale temporal features as input, thereby obtaining the embedded vectors. Meanwhile, it calculates the Euclidean distance between the corresponding elements of the embedded vector and prototype to get the distance features. As a final step, AFFM fuses the multi-scale temporal and distance features to obtain the fused features, based on which classification is performed.

3.2. Multi-scale dynamic convolutional network

Dynamic convolution can dynamically adjust the convolutional kernels for the input data to extract more expressive features. Thus, it is a better convolution operation than ordinary convolution. However, the traditional dynamic convolution has

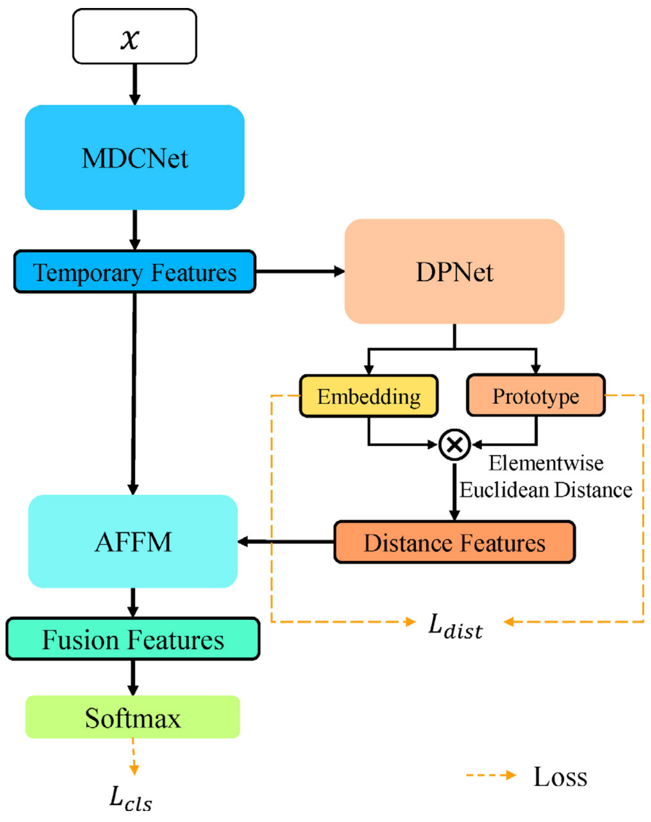

Fig. 1. Architecture of adaptive feature fusion network.

only one size of convolution kernels and cannot extract multi-scale features. Therefore, to extract the multi-scale features of time series, we design a multi-scale dynamic convolution network with multiple sizes of convolution kernels. Thus, it not only retains the advantages of dynamic convolution, but also can capture multi-scale temporal features of time series.

Owing to ResNet's outstanding performance on TSC tasks [15], we build a multi-scale dynamic convolution network (MDCNet) based on ResNet. Its architecture is depicted in Fig. 2. Specifically, for a given input, the MDCNet first uses three stacked blocks to extract features and then uses global average pooling to pool the features, followed by a final Softmax-based classification. Each block consists of multi-scale dynamic convolution residual block (MDC ResBlock), max pooling, SE block and Dropout. The MDC ResBlock is used for extracting multi-scale information. The max pooling, which has a step size of 2, can halve the input data length to reduce the computational burden and effectively increase computational speed. The SE block applies an attention mechanism to the feature maps so that the model can pay attention to more prominent features. Finally, the impact of model overfitting is mitigated by Dropout. We will describe the multi-scale dynamic convolution and the MDC ResBlock below.

3.2.1. Multi-scale dynamic convolution

Dynamic convolution can dynamically adjust the convolution kernel parameters for different inputs. To take advantage of dynamic convolution and extract multi-scale features of time series, we propose multi-scale dynamic convolution (MDC) based on dynamic convolution, whose architecture is depicted in Fig. 3.

It is clear from Fig. 3 that for given m different sizes of convolution kernels $\boldsymbol{W} = \{\boldsymbol{w}_1, \boldsymbol{w}_2, \dots, \boldsymbol{w}_m\}$, the number of convolution kernels $\boldsymbol{w}_i$ of each size is n ($n = 3$ in Fig. 3), that is, $\boldsymbol{w}_i = \{\boldsymbol{w}_i^1, \boldsymbol{w}_i^2, \dots, \boldsymbol{w}_i^n\}$. In the formula, the dimension of each convolution kernel $\boldsymbol{w}_i^j$ is $[C_{out}, C_{in}, k]$, where k denotes the convolution kernel size, so the parameter dimension of convolution kernels for MDC is $[m \times n, C_{out}, C_{in}, k]$. The use of multiple sizes of convolution kernels can effectively capture the multi-scale features of data, remedy the shortcoming of single-scale information with dynamic convolution, and further improve the model performance.

Regarding the operational process of MDC, given an input time series $\boldsymbol{x} \in R^{1\times T}$, where T denotes the series length. Initially, a set of weights $\{\alpha_i^j\}$ is calculated through the attention mechanism, where α_i^j is the weight of corresponding convolution kernel $\boldsymbol{w}_i^j$. By a weighted summation of all convolution kernels, the convolution kernel $\boldsymbol{W}$ is derived, and then convolution operation is performed on the input to obtain the output. The computational is given by

$$MDC(\boldsymbol{x}) = \left(\sum_{i=1}^{m} \sum_{j=1}^{n} \alpha_i^j \boldsymbol{w}_i^j \right) * \boldsymbol{x} \tag{1}$$

where "$*$" stands for the convolution operation.

Since the generation of weights $\{\alpha_i^j\}$ depends on the input data, the model needs to adjust the convolution kernels based on the weights according to different inputs. To get the appropriate weights $\{\alpha_i^j\}$, global average pooling (GAP) is first used to integrate information in the channel direction, and then the convolution operation with a kernel size of 1 and channel number of $m \times n$ is performed to obtain temporal attention map. Temporal attention emphasizes changes of features in the time dimension, i.e. the content information of the series itself. Based on this information, different data can be better distinguished. To obtain $m \times n$ weights, GAP is used to compress the dimension of temporal attention map from $[m \times n, T]$ to $[m \times n, 1]$. The calculation of this process can be described as

$$Attention(\boldsymbol{x}) = Sigmoid(GAP(\boldsymbol{w}_a * GAP(\boldsymbol{x}))) \tag{2}$$

3.2.2. MDC residual block

As shown in Fig. 2, the multi-scale dynamic convolution residual block consists of MDC, Batch Normalization and PReLU activation functions. This block is calculated as follow:

$$\boldsymbol{h} = PReLU(BN(\boldsymbol{W}_1 * \boldsymbol{x} + \boldsymbol{b}_1)) \tag{3}$$

$$\boldsymbol{o} = PReLU(BN(\boldsymbol{W}_2 * \boldsymbol{h} + \boldsymbol{b}_2) + \boldsymbol{x}) \tag{4}$$

where $\boldsymbol{W}_1$ and $\boldsymbol{W}_2$ represent convolution kernels of two MDC in Fig. 2 respectively, $\boldsymbol{b}_1$ and $\boldsymbol{b}_2$ represent the biases of two MDC respectively, $BN\,(\cdot)$ is Batch Normalization, and $PReLU\,(\cdot)$ is the PReLU activation function.

3.3. Distance prototype network

Distance feature is another commonly used feature for time series classification, mainly measuring the distance between time series, and very convenient to extract. By combining distance features with multi-scale temporal features, we are expected to get more comprehensively characteristics of time series data. Therefore, we propose a distance prototype network to extract the distance features of time series. Firstly, the multi-scale temporal features extracted by MDCNet are mapped to the embedding space. Then, the distance between the embedded vector and the prototype is calculated to extract the distance features. Here prototype is a learnable parameter in the distance prototype network, and each prototype represents a category in the dataset.

What the MDCNet extracts are the multi-scale temporal features of time series. Such features focus only on a single time series without considering the heterogeneity between different

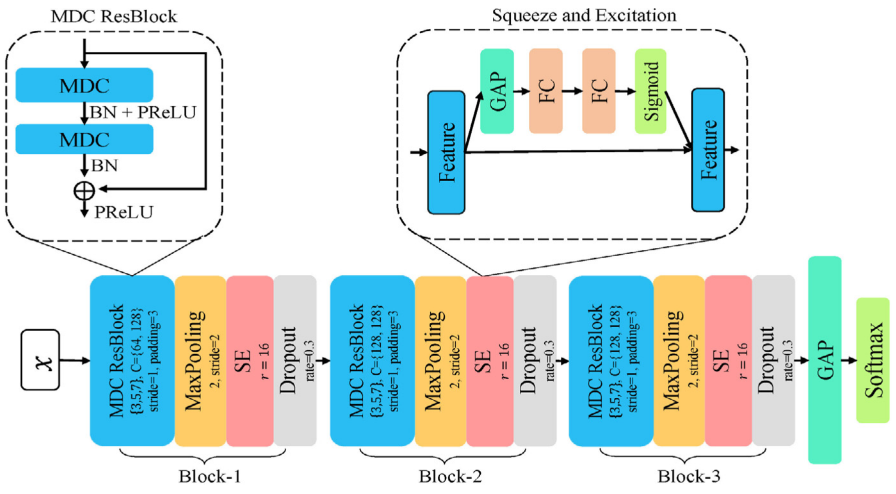

Fig. 2. Architecture of multi-scale dynamic convolution network.

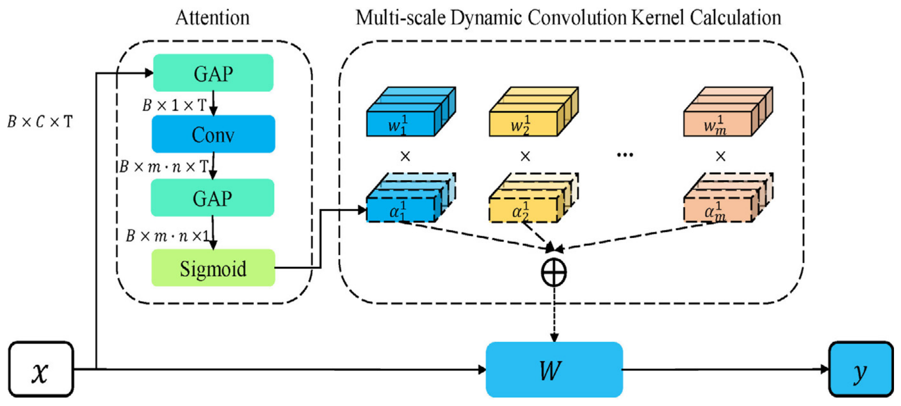

Fig. 3. Multi-scale dynamic convolution.

time series. Since the distance features measure the degrees of similarity and difference between time series, this paper proposes DPNet, a distance prototype network, to extract the distance features of time series. By using distance features as a supplement to the multi-scale temporal features, richer and more discriminative features can be obtained. Fig. 4 presents the architecture diagram of DPNet.

As shown in Fig. 4, the DPNet comprises two fully-connected layers and a learnable prototype parameter $\boldsymbol{P} \in R^{S\times D}$, where S denotes the number of categories and D denotes the feature dimension. Taking the temporal features $\boldsymbol{F}_t \in R^{1\times D}$ extracted by the MDCNet as input, the DPNet first obtains the embedded vector $\boldsymbol{e} \in R^{1\times D}$ via the fully-connected layers, given by

$$\boldsymbol{e} = \boldsymbol{W}_{p2} ReLU(\boldsymbol{W}_{p1}\boldsymbol{F}_t) \tag{5}$$

where $\boldsymbol{W}_{p1} \in R^{\frac{D}{4}\times D}$, and $\boldsymbol{W}_{p2} \in R^{D\times \frac{D}{4}}$.

Then, by element-wise calculation of the Euclidean distance between the embedded vector $\boldsymbol{e}$ and each prototype $\boldsymbol{p}$, the distance features $\boldsymbol{F}_d = (\boldsymbol{f}_1, \boldsymbol{f}_2, \dots, \boldsymbol{f}_S)^T$ of time series can be obtained. In the formula, $\boldsymbol{f}_s \in R^{1\times D}$ denotes the element-wise distance vector between $\boldsymbol{e}$ and a prototype $\boldsymbol{p}_s$ of category s, which is defined as:

$$\boldsymbol{f_s} = \left(\left\|\boldsymbol{e_1} - \boldsymbol{p}_{s,1}\right\|_2, \left\|\boldsymbol{e}_2 - \boldsymbol{p}_{s,2}\right\|_2, \dots, \left\|\boldsymbol{e}_D - \boldsymbol{p}_{s,D}\right\|_2\right) \tag{6}$$

where $\|\cdot\|_2$ represents the Frobenius norm.

3.4. Adaptive feature fusion

Assuming the multi-scale temporal feature extracted by MDC-Net is $\boldsymbol{F}_t$ and the distance feature extracted by DPNet is $\boldsymbol{F}_d$, then $\boldsymbol{F}_t$ represents a kind of high-level semantic information. In contrast, the distance feature represented by $\boldsymbol{F}_d$ is relatively low-level information. Meanwhile, since $\boldsymbol{F}_d$ reflects the distance between the embedded space representation $\boldsymbol{e}$ of $\boldsymbol{F}_t$ and the prototype $\boldsymbol{p}$, there is a certain relationship between $\boldsymbol{F}_t$ and $\boldsymbol{F}_d$. Nevertheless, there are also differences between the two, which provides a certain prerequisite for feature fusion. The fusion of two uncorrelated features possibly makes the fused features meaningless due to the domain disparity, interfering with model training. Besides, it is redundant to merge two types of features that are excessively related. Hence, we consider that by fusing features $\boldsymbol{F}_t$ and $\boldsymbol{F}_d$ through a proper method, features with better expressiveness and discriminativeness can be obtained. To achieve this goal, AFFM is proposed, whose architecture is depicted in Fig. 5.

Inspired by Zhou et al. [48], we develop a feature fusion architecture as displayed in Fig. 5(a). It aims to achieve alternate enhancement by alternately utilizing two types of feature information to obtain fused features. In other words, the information contained in one feature is used to enhance the other feature, and vice versa. Thus, the two features are alternately improved in this way to obtain the fused features ultimately. To realize the “enhancement” process, we present the Fusion Module t (FMt) and the Fusion Module d (FMd), whose architectures are respectively depicted in Fig. 5(b) and (c). As is clear, the two

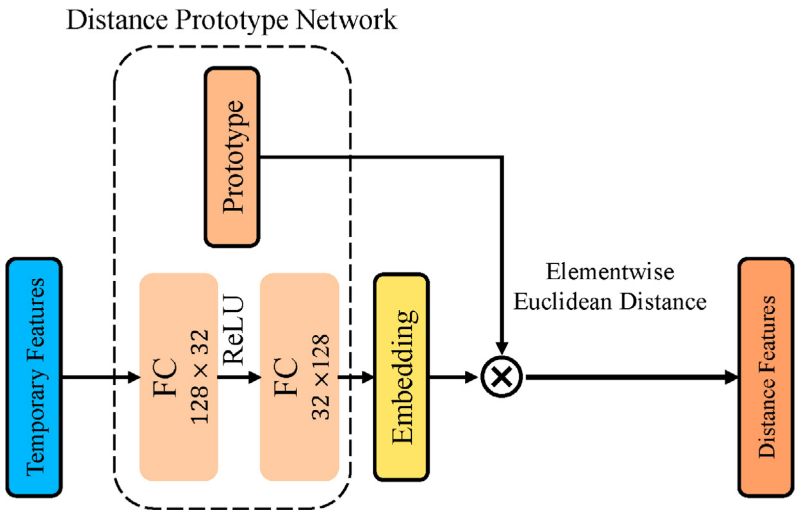

Fig. 4. The flow of distance prototype network architecture.

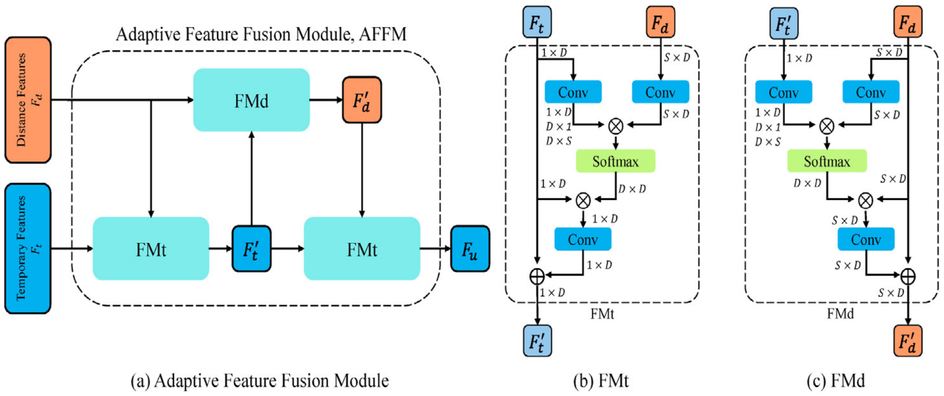

Fig. 5. Adaptive feature fusion module.

modules are structurally similar, and the primary purpose of using these two similar modules is to address the dimensional inconsistency between the features $\boldsymbol{F}_t$ and $\boldsymbol{F}_d$. The design idea of FMt and FMd is to calculate the similarity between the two feature positions, based on which the features are selected and then enhanced. With this approach, similar feature parts can be more easily fused to attain better feature representations. Meanwhile, fusing distinct feature parts may yield meaningless feature representations, which affect the model efficiency.

Specifically, we consider that the multi-scale temporal feature $\boldsymbol{F}_t$ is a high-level feature and should dominate the feature fusion since it is more likely to attain good results. As shown in Fig. 5(a), the AFFM first uses the FMt to fuse the information of $\boldsymbol{F}_d$ into $\boldsymbol{F}_t$ to obtain $\boldsymbol{F}'_t$, and then uses the FMd to fuse the information of $\boldsymbol{F}'_t$ into $\boldsymbol{F}_d$ to obtain $\boldsymbol{F}'_d$. Finally, the FMt is used again to fuse $\boldsymbol{F}'_t$ and $\boldsymbol{F}'_d$, thereby obtaining the fused feature $\boldsymbol{F}_u$. This interaction process can be described as

$$\boldsymbol{F}'_t = FMt(\boldsymbol{F}_t, \boldsymbol{F}_d) \tag{7}$$

$$\boldsymbol{F}'_d = FMd(\boldsymbol{F}_d, \boldsymbol{F}'_t) \tag{8}$$

$$\boldsymbol{F}_u = FMt(\boldsymbol{F}'_t, \boldsymbol{F}'_d) \tag{9}$$

Next, we describe the AFFM in detail. Taking the FMt as an example, for the input features $\boldsymbol{F}_t \in R^{1\times D}$ and $\boldsymbol{F}_d \in R^{S\times D}$, two convolutions with a kernel size of 1 are first used to separately map $\boldsymbol{F}_t$ and $\boldsymbol{F}_d$ to obtain $\boldsymbol{F}^{\theta}_t \in R^{1\times D}$ and $\boldsymbol{F}^{\varphi}_d \in R^{S\times D}$. That is, $\boldsymbol{F}^{\theta}_t = \boldsymbol{w}_{\theta} * \boldsymbol{F}_t$ and $\boldsymbol{F}^{\varphi}_d = \boldsymbol{w}_{\varphi} * \boldsymbol{F}_d$, where $\boldsymbol{w}_{\theta}$ and $\boldsymbol{w}_{\varphi}$ are the convolutional kernel parameters. Then, the similarity between the two features is calculated through matrix multiplication to derive the similarity matrix $\boldsymbol{M}_s \in R^{D\times D}$, $\boldsymbol{M}_s = (\boldsymbol{F}^{\theta}_t)^T\boldsymbol{F}^{\varphi}_d$. $\boldsymbol{M}_s$ contains the correlation degree information of features $\boldsymbol{F}^{\theta}_t$ and $\boldsymbol{F}^{\varphi}_d$ at various positions. By the combined effect of $\boldsymbol{F}^{\theta}_t$ and $\boldsymbol{F}^{\varphi}_d$, $\boldsymbol{M}$ is used to perform $\boldsymbol{F}_t$ selection, thereby selecting beneficial information, i.e. $\boldsymbol{F}^{s}_t = \boldsymbol{F}_t Softmax(\boldsymbol{M}_s)$, $\boldsymbol{F}^{s}_t \in R^{1\times D}$, where the $Softmax(\cdot)$ is to normalize $\boldsymbol{M}_s$. Finally, the feature information is adjusted through convolution and then summed with $\boldsymbol{F}_t$ to derive the preliminary fused feature $\boldsymbol{F}'_t = \boldsymbol{w}_w * \boldsymbol{F}^{s}_t + \boldsymbol{F}_t$. Similarly, the feature $\boldsymbol{F}'_d$ can be obtained by FMd, and the final fused feature $\boldsymbol{F}_u$ can be derived by inputting $\boldsymbol{F}'_t$ and $\boldsymbol{F}'_d$ again into the FMt.

3.5. Complexity analysis

In this subsection, we will briefly analyze the model complexity of the proposed method. The complexity of AFFNet mainly comes from three parts: multi-scale dynamic convolution, distance prototype network and adaptive feature fusion module. The first part is multi-scale dynamic convolution, for an input feature map with dimension $C_{in} \times T$, where C_{in} is the number of input channel, T is the length of sequence. MDC needs to computes attention weights and its complexity is about $(C_{in} + T + mnT)$ multiply-adds, where mn is the number of convolution kernel used in MDC. Therefor, the computational cost of convolution kernel summation is $(mnC_{in}C_{out}k + mnC_{out})$ multiply-adds, where

k is the maximum convolutional kernel size. Accordingly, the complexity of the convolution calculation is $(kC_{in}C_{out}T)$ multiply-adds. The second part is distance prototype network, the fully-connected layer requires about D^2 multiply-adds, where D stands for feature dimension. And the cost of distance calculation is around SD multiply-adds, where S is the number of classes. The third part is AFFM, AFFM requires three convolution calculations and two matrix multiplication operations in one interaction process, so the computational cost of Fusion Module t (FMt) is about $3C_{in}C_{out}D + SD^2 + D^2$.

For example, when the length of the time series is 150, if all multi-scale convolution kernels of AFFNet are set to $\boldsymbol{W} = \{3, 5, 7\}$, the number of each kernel size n is 3 and the feature dimension is set as $D = 128$, The FLOPs and the number of parameters of AFFNet will be 30.78 Million and 7.8 Million. However, if we use ordinary convolution with kernel size of 7 in AFFNet, its FLOPs and the number of parameters will be 26.90 Million and 0.5 Million. It can be seen that our model has a greater number of parameters.

3.6. Loss function

The proposed AFFNet performs end-to-end training. We use Mixup [49] technology in the training process. In Mixup, we randomly select two samples $(\boldsymbol{x}_i, \boldsymbol{y}_i)$ and $(\boldsymbol{x}_j, \boldsymbol{y}_j)$ each time, where $\boldsymbol{x}_i$ and $\boldsymbol{x}_j$ represent the raw time series data, $\boldsymbol{y}_i$ and $\boldsymbol{y}_j$ are their corresponding one-hot labels, then we calculate the new sample $(\tilde{\boldsymbol{x}}, \tilde{\boldsymbol{y}})$ by linear interpolation:

$$\tilde{\boldsymbol{x}} = \lambda \boldsymbol{x}_i + (1-\lambda)\boldsymbol{x}_j, \tag{10}$$

$$\tilde{\boldsymbol{y}} = \lambda \boldsymbol{y}_i + (1-\lambda)\boldsymbol{y}_j, \tag{11}$$

where λ is a random number from $Beta(\beta, \beta)$ distribution. By using this augmentation method, we not only increase the coverage space of the training data, but also reduce the inadaptability of the model in predicting data, increasing the uncertainty estimation ability of the model, and thus improving the generalization ability of the model to a certain extent. While the loss function of AFFNet includes two parts: one is the classification loss L_{cls} used for classification, and the other is the distance loss L_{dist} used to train the prototype parameter $\boldsymbol{P}$ in DPNet. The classification loss L_{cls} uses Label Smooth cross-entropy loss [50], which is defined as

$$L_{cls} = -\left((1-\varepsilon)\sum_{i=1}^{N}\tilde{\boldsymbol{y}}_i \log \hat{\boldsymbol{y}}_i + \frac{\varepsilon}{S}\sum_{i=1}^{N}\log \hat{\boldsymbol{y}}_i\right) \tag{12}$$

where ε is a small constant, we select ε as 0.1 here; $\tilde{\boldsymbol{y}}$ and $\hat{\boldsymbol{y}}$ are true labels and predicted labels respectively, S is the number of categories, and N is the number of training samples.

To obtain an effective prototype representation, we use the distance loss based on the classification to train the prototype parameter $\boldsymbol{P}$ in DPNet, so that the distance between the embedding vector $\boldsymbol{e}$ and the prototype $\boldsymbol{p}$ of the same category is as close as possible. Besides, the distance between the embedding vector $\boldsymbol{e}$ and the prototype $\boldsymbol{p}$ of different categories is as considerable as possible. The loss is defined as follows:

$$p(s|\boldsymbol{e}) = \frac{\exp(-D_f(\boldsymbol{e}, \boldsymbol{p}_s))}{\sum_{i=1}^{S}\exp(-D_f(\boldsymbol{e}, \boldsymbol{p}_i))} \tag{13}$$

$$L_{dist} = -\sum_{i=1}^{N}\log p(s|\boldsymbol{e}_i) \tag{14}$$

where $D_f(\cdot)$ denotes the Euclidean distance, $p(s|\boldsymbol{e})$ represents the probability that the embedding vector $\boldsymbol{e}$ belongs to class s, and N is the number of training samples. Therefore, the final loss is:

$$L = L_{cls} + \gamma L_{dist} \tag{15}$$

where L_{cls} is the classification loss, L_{dist} is the distance loss and γ is a trade-off factor.

During training, the classification loss and the distance loss work together to guide the network to convergence. Therefore, we can optimize all the parameters simultaneously.

4. Experimental results

In this section, we first introduce the datasets used in the experiment. Secondly, we present the implementation details and evaluation indicators. Thirdly, we compare our model with other models to verify the superiority of the proposed method. Finally, we carry out the ablation experiments to analyze the influence of parameters on the AFFNet. The experiments were conducted in Intel (R) Xeon (R) e5-2603 V4, 1.70 GHz CPU, 128G RAM and NVIDIA Tesla k40c 12G GPU. We provided our code in Github, and it is available at: https://github.com/Damienwang20/adaptive-feature-fusion.

4.1. Datasets

To verify the performance of the proposed model, we select 85 UCR datasets [51] to perform experiments. Additionally, to explore its performance on multivariable time series, we select 3 human activity recognition datasets (WISDM [52], HAR [53] and Opportunity [54]) to conduct experiments. In this subsection, we sequentially describe the UCR, WISDM, HAR and Opportunity datasets.

The UCR [51] dataset contains a variety of types of univariate time series datasets, such as ECG, sensor, image, etc. These datasets also have a wide range of lengths and categories, varying from 15 to 2844 and 2 to 60, respectively. At present, there are 128 datasets in UCR. To facilitate the comparison with other models, we select 85 datasets for experiments in this paper. Since these datasets have been divided into training sets and test sets, we will use the divided training sets and test sets for experiments. Details of 85 dataset are given as Table 1.

The WISDM [52] dataset collected sensor data from 36 individuals performing six types of actions, including walking, jogging, climbing stairs, sitting, lying and standing. According to literature, [55], 1–26 are used as the training set, 27–36 as the test set in this paper. The data recorded 20 points in one second. The data fragments are intercepted with an interval of 1 s and a window size of 10 s. Thus, a sequence sample is generated with a length of 200. After processing, the training set size of the data is 7367, the test set size is 3026, the sequence length is 200, the number of channels is 3, and there are 6 categories.

HAR [53] dataset consists of data information of six kinds of actions performed by 30 people, including walking, going up and downstairs, sitting, standing and lying down. According to [56], the data recorded in 2.56 s is used as a sample, i.e., the sequence length of a sample is 128. In this way, there are 7406 training samples, 2993 test samples and 6 data sensor channels in the dataset. Thus, there are 6 categories in total.

Opportunity [54] is one of the baseline datasets for human activity recognition, collected data from four people performing 17 activities in the kitchen, each of which was performed 6 times. Those data are stored in 5 ADL files and 1 Drill file. To compare with other methods, we processed the data according to the method in literature [56], and selected the activity data of 3 people and each person who performed 6 times. The test set was composed of the fourth and fifth activity data of No. 2 and No. 3, and the rest of the data were composed of the training set. For missing values in the data, this paper uses linear interpolation padding and Z-score normalized the data. After processing, 46 495 training samples and 9894 test samples were obtained. The dataset contains 17 categories and NULL class, i.e. 18 categories, the sequence length is 24, and the number of sensor channels is 113.

Table 1
Information of 85 UCR time series datasets.

Datasets	Train	Test	Class	Length	Type
Adiac	390	391	37	176	Image
ArrowHead	36	175	3	251	Image
Beef	30	30	5	470	Spectro
BeetleFly	20	20	2	512	Image
BirdChicken	20	20	2	512	Image
Car	60	60	4	577	Sensor
CBF	30	900	3	128	Simulated
Chlorine	467	3840	3	166	Sensor
CinCECGTorso	40	1380	4	1639	Sensor
Coffee	28	28	2	286	Spectro
Computers	250	250	2	720	Device
CricketX	390	390	12	300	Motion
CricketY	390	390	12	300	Motion
CricketZ	390	390	12	300	Motion
DiatomSizeR	16	306	4	345	Image
DisPhxAgeGp	400	139	3	80	Image
DisPhxCorr	600	276	2	80	Image
DisPhxTW	400	139	6	80	Image
Earthquakes	322	139	2	512	Sensor
ECG200	100	100	2	96	ECG
ECG5000	500	4500	5	140	ECG
ECGFiveDays	23	861	2	136	ECG
ElectricDevices	8926	7711	7	96	Device
FaceAll	560	1690	14	131	Image
FaceFour	24	88	4	350	Image
FacesUCR	200	2050	14	131	Image
FiftyWords	450	455	50	270	Image
Fish	175	175	7	463	Image
FordA	3601	1320	2	500	Sensor
FordB	3636	810	2	500	Sensor
GunPoint	50	150	2	150	Motion
Ham	109	105	2	431	Spectro
HandOutlines	1000	370	2	2709	Image
Haptics	155	308	5	1092	Motion
Herring	64	64	2	512	Image
InlineSkate	100	550	7	1882	Motion
InsectWing	220	1980	11	256	Sensor
ItalyPower	67	1029	2	24	Sensor
LrgKitApp	375	375	3	720	Device
Lightning2	60	61	2	637	Sensor
Lightning7	70	73	7	319	Sensor
Mallat	55	2345	8	1024	Simulated
Meat	60	60	3	448	Spectro
MedicalImages	381	760	10	99	Image
MidPhxAgeGp	400	154	3	80	Image
MidPhxCorr	600	291	2	80	Image
MidPhxTW	399	154	6	80	Image
MoteStrain	20	1252	2	84	Sensor
NonInvThor1	1800	1965	42	750	ECG
NonInvThor2	1800	1965	42	750	ECG
OliveOil	30	30	4	570	Spectro
OSULeaf	200	242	6	427	Image
PhalCorr	1800	858	2	80	Image
Phoneme	214	1896	39	1024	Sensor
Plane	105	105	7	144	Sensor
ProxPhxAgeGp	400	205	3	80	Image
ProxPhxCorr	600	291	2	80	Image
ProxPhxTW	400	205	6	80	Image
RefrigerationDevices	375	375	3	720	Device
ScreenType	375	375	3	720	Device
ShapeletSim	20	180	2	500	Simulated
ShapesAll	600	600	60	512	Image
SmlKitApp	375	375	3	720	Device
SonyAIBORobot1	20	601	2	70	Sensor
SonyAIBORobot2	27	953	2	65	Sensor
StarLightCurves	1000	8236	3	1024	Sensor
Strawberry	613	370	2	235	Spectro
SwedishLeaf	500	625	15	128	Image
Symbols	25	995	6	398	Image
SyntheticControl	300	300	6	60	Simulated
ToeSegmentation1	40	228	2	277	Motion
ToeSegmentation2	36	130	2	343	Motion
Trace	100	100	4	275	Sensor
TwoLeadECG	23	1139	2	82	ECG
TwoPatterns	1000	4000	4	128	Simulated
UWaveGestAll	896	3582	8	945	Motion
UWaveGestX	896	3582	8	315	Motion
UWaveGestY	896	3582	8	315	Motion
UWaveGestZ	896	3582	8	315	Motion
Wafer	1000	6164	2	152	Sensor
Wine	57	54	2	234	Spectro
WordSynonyms	267	638	25	270	Image
Worms	181	77	5	900	Motion
WormsTwoClass	181	77	2	900	Motion
Yoga	300	3000	2	426	Image

(*continued on next page*)

Table 1 (*continued*).

4.2. Implementation details

We use cross-validation on the standard training set to obtain training parameters: the trade-off factor γ and the number of convolutional kernels of each size n. In particular, we first split the standard training set into two halves, one new half for training and the other for testing. Then, we use 2-fold cross-validation on the training set to select γ in {0.5,1,1.5,2.0} and n in {1,2,3,5,7}. Finally, we retrain the model with the standard training set and the standard testing set, and report their results. The exact values of γ is shown in Table 2, and n is 2.

During training, the convolutional kernel parameters $\boldsymbol{W}$ of all multi-scale dynamic convolutions in AFFNet are set to $\boldsymbol{W} = \{3, 5, 7\}$, the stride is 1. The max-pooling kernel size is 2, and the stride is 2. The reduction in SE is set to 16. The drop rate in dropout is 0.3. For the input sequence $\boldsymbol{x}$ with length T, after calculation, its length will be reduced by 8 times, and finally, the temporal feature vector with the dimension of 1×128 is obtained. The number of hidden layer nodes of two FC layers in DPNet is 32 and 128, respectively. For datasets with S categories, the initialization dimension of parameter $\boldsymbol{P}$ is $[S, 128]$, and the distance feature extracted by DPNet is a feature vector of $S \times 128$ dimension. In this paper, each dataset is trained at 600 epochs. The batch size is 64, and β is set to 0.2. The learning rate is set to 0.01. Starting from the 200th iteration, the learning rate decreases by 0.5 times every 100 iterations. Adam algorithm [57] is used to optimize the loss function.

For the WISDM dataset, we set the batch size to 128 and train 1000 epochs in total. The DPNet learning rate in AFFNet is set to 0.03, the learning rate of the rest part of AFFNet is 0.01, and the value of γ is 1.0. From the 200th iteration, the learning rate decreases by 0.5 times for every 100 iterations and remains unchanged after the 600th iteration. Other hyperparameters are consistent with UCR settings.

For HAR dataset, the convolutional kernel parameters $\boldsymbol{W}$ of all multi-scale dynamic convolutions in AFFNet are set to $\boldsymbol{W} = \{3, 5, 7\}$, and n is 5. The remaining hyperparameters are consistent with the settings of WISDM.

For Opportunity dataset, the convolutional kernel parameters are set to $\boldsymbol{W} = \{3, 5, 7\}$, n is 2. The batch size is set to 512, a total of 150 epochs of training, the DPNet learning rate in AFFNet is set to 0.05, the rest of the learning rate is set to 0.01, and the value of γ is 0.5. The learning rate decreased by 0.5 times every 30 iterations and remained unchanged after the 90th iteration. Other hyperparameters are the same as those set in UCR.

4.3. Comparison with other models

4.3.1. Comparison methods

For the UCR datasets, according to Ref. [58], we selected several representative methods for comparison. They are given below.

Table 2
Value of Gamma for each dataset.

Datasets	γ	Datasets	γ	Datasets	γ
Adiac	2.0	FordB	0.5	RefrigerationDevices	1.5
ArrowHead	0.5	GunPoint	0.5	ScreenType	1.0
Beef	1.5	Ham	1.0	ShapeletSim	0.5
BeetleFly	2.0	HandOutlines	2.0	ShapesAll	0.5
BirdChicken	1.0	Haptics	1.5	SmlKitApp	1.0
Car	0.5	Herring	0.5	SonyAIBORobot1	2.0
CBF	0.5	InlineSkate	0.5	SonyAIBORobot2	0.5
Chlorine	0.5	InsectWing	2.0	StarLightCurves	1.5
CinCECGTorso	1.0	ItalyPower	1.5	Strawberry	1.5
Coffee	0.5	LrgKitApp	0.5	SwedishLeaf	0.5
Computers	1.0	Lightning2	2.0	Symbols	1.0
CricketX	2.0	Lightning7	1.0	SyntheticControl	1.0
CricketY	0.5	Mallat	1.0	ToeSegmentation1	1.5
CricketZ	1.0	Meat	2.0	ToeSegmentation2	1.0
DiatomSizeR	0.5	MedicalImages	1.5	Trace	0.5
DisPhxAgeGp	2.0	MidPhxAgeGp	0.5	TwoLeadECG	1.0
DisPhxCorr	1.0	MidPhxCorr	1.0	TwoPatterns	0.5
DisPhxTW	0.5	MidPhxTW	0.5	UWaveGestAll	0.5
Earthquakes	1.5	MoteStrain	1.0	UWaveGestX	2.0
ECG200	0.5	NonInvThor1	1.5	UWaveGestY	2.0
ECG5000	1.0	NonInvThor2	2.0	UWaveGestZ	2.0
ECGFiveDays	0.5	OliveOil	1.5	Wafer	1.5
ElectricDevices	1.5	OSULeaf	0.5	Wine	2.0
FaceAll	2.0	PhalCorr	1.0	WordSynonyms	1.0
FaceFour	1.0	Phoneme	1.5	Worms	2.0
FacesUCR	0.5	Plane	0.5	WormsTwoClass	0.5
FiftyWords	1.5	ProxPhxAgeGp	1.0	Yoga	2.0
Fish	0.5	ProxPhxCorr	2.0		
FordA	2.0	ProxPhxTW	2.0		

Distance-based methods: 1NN with Euclidean Distance(ED) [5], 1NN with Dynamic Time Warping (DTW) [5], Move–split–merge (MSM) [59];

Feature-based methods: learned shapelet (LS) [26], fast shapelet (FS) [25], bag of SFA symbols (BOSS) [23], time series forest (TSF) [6], Time series bag of features (TSBF) [60], Learned pattern similarity (LPS) [61];

Ensemble-based methods: shapelet transform (ST) [62], elastic ensemble (EE) [27], the collective of transformation-based ensembles (COTE) [28];

Deep learning-based methods: Multilayer Perceptron (MLP) [31], Fully Convolutional Network (FCN) [31], Residual Network (ResNet) [31], Echo Memory-Augmented Network (EMAN) [36], TSC-FF [20], OS-CNN [18], DMS-CNNs [44]

In this experiment, Wins, Mean Accuracy, Avg Rank and MPCE [31] are used as the evaluation indicators of the models on UCR datasets. Among them, Wins indicates how many datasets the model have achieved the best performance, including the number of tie results. That is to say, if multiple models achieved the highest accuracy on a dataset at the same time, this dataset will be counted in wins for all these models. Therefore, the sum of wins of all the models is greater than the number of datasets, i.e., 85. Mean accuracy represents the average accuracy of the model on all datasets. AVG Rank is the mean of the classification accuracy ranking over the 85 UCR datasets. Finally, MPCE represents Mean Per-Class Error, i.e., the average error of each class. The PCE and MPCE can be calculated as:

$$PCE_a = \frac{1 - accuracy}{number\ of\ classes} \tag{16}$$

$$MPCE = \frac{1}{A}\sum_a PCE_a \tag{17}$$

where *number of classes* means the number of categories in a dataset, and A is the number of datasets.

For WISDM, HAR and Opportunity dataset, we chose the following 9 methods for comparison. They are 40 statistical features and Random Forest (RF) [55], PCA features and RF [55], raw sequence and kNN [55], 40 statistical features and CNN [55], RF [63] and Dropout classifier using artificially designed feature [63], classification using CNN [64], CNN [65] and residual bidirectional LSTM (Res-Bidir-LSTM) network [56]. Accuracy is taken as the evaluation index on these three datasets.

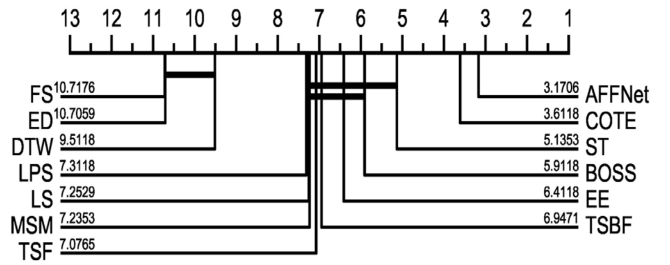

Fig. 6. Critical difference diagram of the comparison with traditional methods on UCR datasets.

4.3.2. Comparison results

Tables 3, 4 detail the comparison results of AFFNet with other methods on UCR, WISDM, HAR and Opportunity datasets. Additionally, we make significant comparisons on the UCR datasets by Wilcoxon Signed-Rank test [15], and the results are shown in Figs. 6 and 7, where the models connected by horizontal lines indicate the absence of significant differences between them.

As is clear from Tables 3, 4 and Figs. 6, 7, we can get that:

1. AFFNet attains the optimal results regarding Wins, Mean, Avg Rank and MPCE indicators (detailed accuracy data can be found in Table 8). Precisely, AFFNet achieves the highest average accuracy of 0.851 and an average ranking of 5.494. Additionally, AFFNet and FCN concurrently perform the lowest MPCE of 0.0395. Meanwhile, AFFNet attains the optimal results on 22 datasets. This demonstrates that by adaptively fusing two different types of features, AFFNet can achieve the highest performance on time series classification.
2. Compared with traditional methods, the results of our model have more advantages. Moreover, AFFNet is significantly better than all traditional models, including COTE which integrates 35 classifiers. This shows that as a deep learning method, AFFNet can automatically extract two different types of features, and adaptively fuse them. This is helpful for time series classification.
3. Among deep learning methods, FCN, EMAN and other models also get good results, but they can only extract a single style of features. Although DMS-CNNs uses dynamic multi-scale convolution to extract features, its performance is lower than our model. This may be due to the limitation of single type of features. As for TSC-FF, it only uses the concatenating fusion to combine different features, limiting the model performance. The model we proposed, however, can extract multi-scale temporal features and distance features of time series, by effectively fusing these two types of features via the AFFM, AFFNet can obtain more discriminative fused features with stronger expressiveness for classification and achieve better results. Besides, despite insignificant differences from models like EMAN, FCN and TSC-FF, the present method has certain advantages in terms of numerical value.
4. AFFNet attained the highest accuracies on both WISDM and Opportunity datasets, with 0.946 and 0.912, respectively. Besides, it achieved the accuracy of 0.953 on HAR

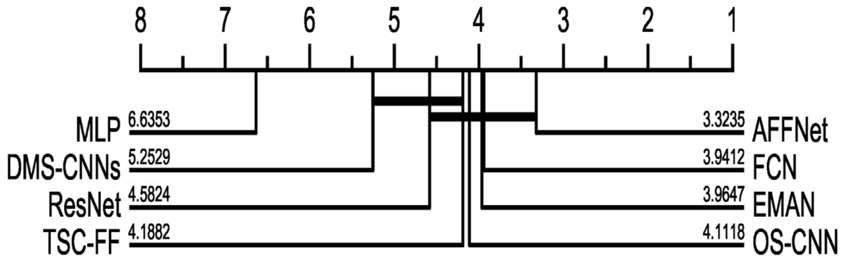

Fig. 7. Critical difference diagram of the comparison with deep learning methods on UCR datasets.

Table 3
Results of multiple models on UCR datasets.

Index	ED	DTW	MSM	LS	FS	BOSS	TSF	TSBF	LPS	ST	EE	COTE	MLP	FCN	ResNet	OS-CNN	TSC-FF	EMAN	DMS-CNNs	AFFNet
Wins	1	4	2	5	3	12	2	4	3	8	6	11	4	19	12	8	16	13	8	**22**
Mean	0.709	0.740	0.796	0.769	0.707	0.810	0.779	0.778	0.795	0.822	0.792	0.838	0.752	0.844	0.838	0.837	0.835	0.843	0.812	**0.851**
Avg Rank	17.065	15.529	12.241	12.282	16.894	10.247	12.206	11.965	12.171	9.200	11.359	6.953	13.271	6.612	7.641	6.947	6.765	6.176	8.982	**5.494**
MPCE	0.0807	0.0734	0.0573	0.0598	0.0753	0.0502	0.0616	0.0591	0.0555	0.0467	0.0598	0.0445	0.0681	**0.0395**	0.0416	0.0449	0.0439	0.0402	0.0487	**0.0395**

Table 4
Comparison results of human activity recognition datasets.

Method	WISDM	HAR	Opportunity
Handcrafted features + RF [63]	0.835	0.778	0.824
Handcrafted features + Dropout [63]	0.854	0.763	0.837
PCA+RF [55]	0.753	0.892	0.867
Raw data + KNN [55]	0.662	0.730	0.879
Basic features + RF [55]	0.827	0.817	0.873
CNN+stat. Features [55]	0.933	**0.976**	0.906
CNN [64]	0.918	0.948	0.901
CNN [65]	0.910	0.921	0.889
Res-Bidir-LSTM [56]	0.872	0.936	0.898
AFFNet	**0.946**	0.953	**0.912**

dataset, which was a slightly lower than that of CNN + Stat.features, while greater than the other 8 methods. This indicates that AFFNet can also achieve good results in multivariate time series datasets.

5. Among the 22 wining datasets, the datasets with less than 300 training samples account for 68% of the total, and the datasets with imbalanced training and test set take possession of 60%. This shows that the proposed network has more advantages in the small and imbalanced datasets.
6. In conclusion, AFFNet extracts two different types of features of time series, which can more accurately describe time series data and improve the classification performance.

4.4. Ablation experiments

4.4.1. Effect of MDC kernels on model performance

To fully investigate the effect of multiple scales on the model performance, we compare the results between ordinary convolution, dynamic convolution and multi-scale convolution on the WISDM, HAR and some UCR datasets. We select four UCR datasets, and they are ItalyPower, SonyAIBORobot2, FacesUCR, and UWaveGestAll. These four datasets have a significantly imbalanced training and test split. Table 5 lists the convolution kernels of different scales. Initially, the MDC of MDCNet in AFFNet is replaced with ordinary convolution, and the sizes of convolution kernels in the three blocks are set to 7, 5 and 3 concerning the settings of FCN [31], and ResNet [31]. The model results are represented by Ordinary in Fig. 8. The MDC kernels in AFFNet are set to the same size to represent the conventional single-scale dynamic convolution. However, since the MDC kernels $\boldsymbol{W} = \{3, 5, 7\}$ used here contain 3 convolution kernels, that is, an MDC uses 3 convolution kernels when $n = 1$. Hence, for a fair comparison, the kernel size is set to {7,7,7} for the first Block in MDCNet, {5,5,5} for the second Block, and {3,3,3} for the third Block. DC represents relevant results in Fig. 8. Meanwhile, MDC(7) signifies that the MDC kernels are {3,5,7}. Additionally, we attempt to increase the scale types in the convolution kernels, and expand the three-scale convolution kernel {3,5,7} to {3,5,7,9} until it is {3,5,7,9,11,13}. Relevant results are represented by MDC(7), MDC(9), MDC(11) and MDC(13), respectively. To control the model size and training time, we only expand the convolution kernel scale of the first two blocks while keeping that of the third Block unchanged ($\boldsymbol{W} = \{3, 5, 7\}$). The corresponding results are displayed in Fig. 8.

As is clear from Fig. 8, we have

1. In most cases, such as on WISDM, FacesUCR, UWaveGestAll datasets, AFFNet using dynamic convolution is superior to that using ordinary convolution. Meanwhile, AFFNet using MDC is also superior to that using single-scale dynamic convolution. This indicates that the dynamic convolution has a stronger feature extraction capability than ordinary convolution, enabling to extracting more effective temporal features. Furthermore, It also shows that MDC can be used to capture the multi-scale temporal features of time series and attain richer and comprehensive feature representations. Compared to single-scale features, multi-scale features contain richer and more accurate information, which are more suitable for describing changes in time series, thereby improving the model classification ability.
2. When increasing the scales of convolution kernels in MDC, the accuracy of the model actually has not changed much, it basically fluctuated within a certain range, especially in WISDM, HAR and FacesUCR datasets. Additionally, the accuracy of the proposed model on datasets ItalyPower and UWaveGestAll tends to be stable, with almost no change. As for SonyAIBORobot2, the model performance is basically declining when the convolution kernel scale increases. Meanwhile, when the number of convolution kernels n increases, the model performance will be further improved in most cases. However, this will bring more parameters. Therefore, we select the scales of {3,5,7} as the final scales of convolution kernels to extract the multi-scale temporal features.

4.4.2. Prototype visualization

This subsection visualizes the embedding vector and prototype obtained by DPNet. Since both the embedding vector and prototypes are 128-dimensional vectors, we first use the t-SNE algorithm to reduce the vector dimensions to 2. Then, to compare the prototype learning effect by DPNet, we display the

Table 5
Multi scale dynamic convolution kernel parameters.

Block	Ordinary	Single scale dynamic DC	Multi scale dynamic MDC(7)	Multi scale dynamic MDC(9)	Multi scale dynamic MDC(11)	Multi scale dynamic MDC(13)
Block-1	w = 7	W = {7,7,7}	W = {3,5,7}	W = {3,5,7,9}	W = {3,5,7,9,11}	W = {3,5,7,9,11,13}
Block-2	w = 5	W = {5,5,5}	W = {3,5,7}	W = {3,5,7,9}	W = {3,5,7,9,11}	W = {3,5,7,9,11,13}
Block-3	w = 3	W = {3,3,3}	W = {3,5,7}	W = {3,5,7}	W = {3,5,7}	W = {3,5,7}

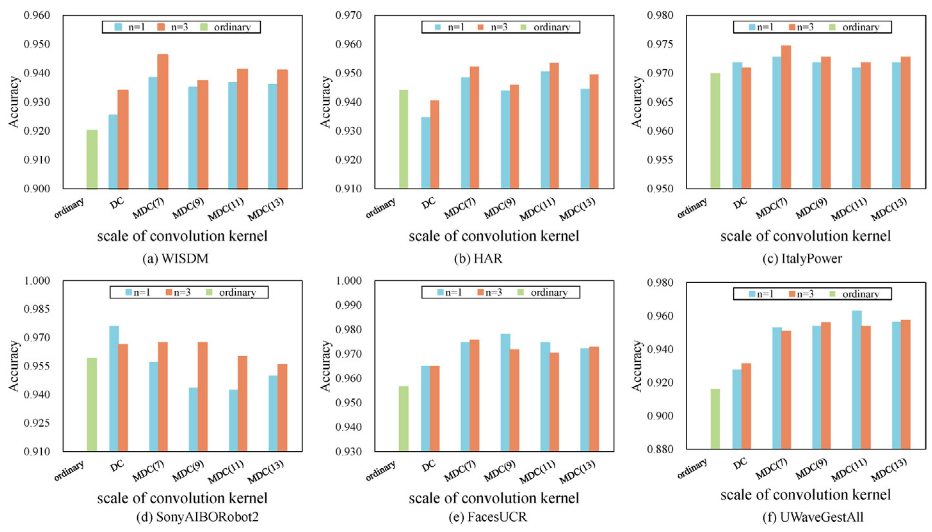

Fig. 8. Influence of convolution kernel scale on the model.

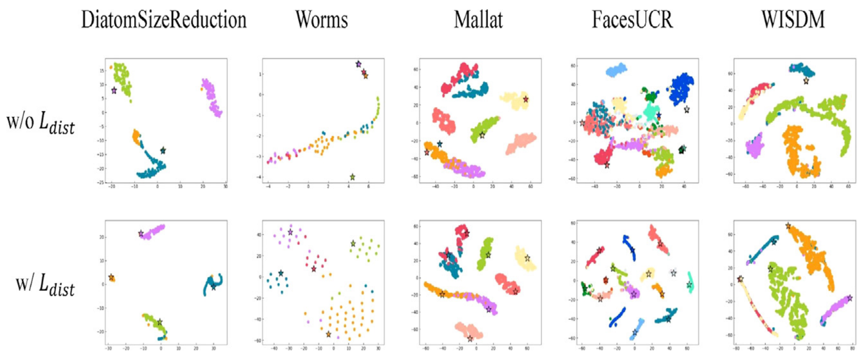

Fig. 9. Prototype visualization.

prototype visualization results in the absence and presence of distance loss in Fig. 9, where "circle" denotes the embedding vector, "star" denotes the prototype, and different colors indicate different categories. As Section 4.4.1 did, we also select four UCR datasets with imbalanced training and test set for visualization, as well as one Human Activity Recognition dataset. They are DiatomSizeReductoin, Worms, Mallat, FacesUCR, and WISDM.

It is clear from Fig. 9 that in the absence of distance loss, the embedded vectors of the same categories are already fundamentally aggregated. However, the prototypes at this time only have random positions. As a result, there are overlaps between different prototypes, especially for the WISDM dataset, where almost all the prototypes are at the same position. Obviously, in this state, the prototypes cannot reflect the category information, and the distance features calculated at this time are meaningless. After incorporating the distance loss, the prototypes representing each category are clustered with their cognate embedded vectors. As a result, their distances from cognate embeddings are shortened, while their distances from embeddings of different categories are widened. For the FacesUCR datasets, in particular, the degree of sample clustering is improved. This fully shows that incorporating distance loss can optimize the prototypes, which thus enables DPNet to obtain effective distance features and prepare for the subsequentfusion [66].

4.4.3. *Effect of weighing coefficient gamma on model performance*

To further investigate the effect of distance loss on the model performance, this subsection also displays the model accuracy variations at different γ values, and Fig. 10 presents the results. As Section 4.4.2 did, we select the same four UCR datasets with imbalanced training and test set for visualization, as well as one Human Activity Recognition dataset. They are DiatomSizeReductoin, Worms, Mallat, FacesUCR, and WISDM, respectively.

According to Fig. 10, which reflects the changes in AFFNet accuracy on 5 datasets at gamma values of {0.0, 0.5, 1.0, 1.5, 2.0, 2.5}, we can find that the variation trends differ among datasets.

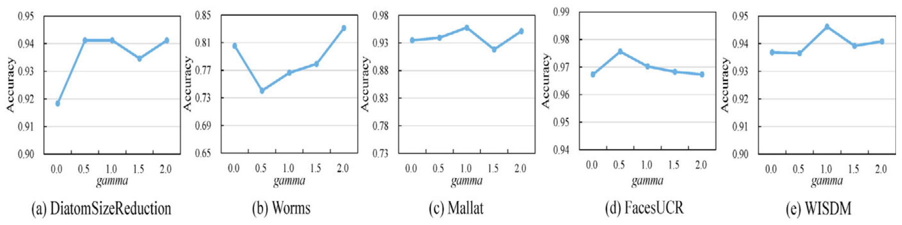

Fig. 10. Influence of gamma value on accuracy.

Table 6
MPCE changes of different fusion modes on 85 UCR datasets.

Model	Fusion method	MPCE
MDCNet	–	0.0430
DPNet	–	0.0446
AFFNet	Add	0.0446
	Concat	0.0453
	Adaptive	**0.0395**

Table 7
Accuracy changes of different fusion methods on three datasets.

Method	Fusion method	WISDM	HAR	Opportunity
MDCNet	–	0.9402	0.9455	0.8997
DPNet	–	0.9068	0.9485	0.8804
AFFNet	Add	0.9399	0.9492	0.9071
	Concat	0.9375	0.9526	0.9043
	Adaptive	**0.9461**	**0.9532**	**0.9120**

For example, in the case of Worms, the accuracy of AFFNet decreases initially and then begins to increase with the increasing γ. The accuracies of the DiatomSizeReduction and Mallat datasets increase first and then fluctuate within a small range. The accuracy on the FacesUCR dataset increases with the increasing gamma and then begins to decrease. As for the WISDM dataset, the accuracy fundamentally fluctuates around 0.94, which peaks at $\gamma = 1.0$. In summary, the value of γ has varying effects on different datasets. In practical applications, it is necessary to select an appropriate γ for the dataset to improve the network performance.

4.4.4. Effect of feature fusion on model performance

To reflect the effectiveness of the AFFM proposed herein, we compared AFFNet with MDCNet, DPNet, as well as with two fusion methods "Add" and "Concat". We used MPCE to evaluate the overall effect of the model on 85 UCR datasets and accuracy to evaluate the effect on the other three datasets. Tables 6 and 7 respectively present the results of different fusion methods on these two types of time series datasets.

As is clear from Tables 6 and 7, we have

1. On the UCR datasets, MDCNet and DPNet respectively achieved the MPCEs of 0.0430 and 0.0466, higher than that of AFFNet with adaptive fusion (0.0395). Besides, on the other three datasets, the accuracy of MDCNet and DPNet are also not as high as that of AFFNet using adaptive fusion. Specifically, compared with MDCNet, AFFNet was 0.59% higher on WISDM, 0.77% higher on HAR and 1.23% higher on Opportunity, respectively. This shows that by fusing different features, we can improve the classification ability of the model.
2. Among the three different fusion strategies, the adaptive fusion style achieved better results than both Add fusion and Concat fusion on all the datasets. It demonstrates that the adaptive feature fusion proposed herein is a more effective method. The reason is that it allows multiple interactions between features so that those features that are beneficial to fusion can be selected according to the similarity information of the two features.
3. In conclusion, AFFNet can effectively fuse the multi-scale temporal and distance features to obtain better feature representations and finally improve the model's classification ability.

5. Conclusions

In this paper, we have proposed a model named Adaptive Feature Fusion Network for time series classification. Our proposed model can adaptively fuse multi-scale temporal features and distance features to classify, which is demonstrated to be more suitable for time series classification. We also presented two networks in the model. One is MDCNet. The other is DPNet. Ablation studies indicated that the proposed networks can effectively extract different features. The comparison experiments showed that the proposed model achieves higher accuracy than the traditional methods and deep learning models on most datasets of UCR and is better than the baseline methods on Human Activity Recognition datasets.

However, the proposed model still has limitations. First, as analyzed in Section 3.5, the number of parameters of the model is large and the computational complexity is high. Second, the distance loss may not be a good choice, because its distance constraint ability is not very strong according to some metric learning research. In future work, more efficient versions should be studied. One probable solution is to prune connections by exploiting the importance of nodes. Second, a new distance loss could be designed to better constraint the relationship between prototypes and embeddings, so as to get a better representation of distance features.

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 is supported by the National Natural Science Foundation of China (61806013, 61876010, 61906005, 62166002), General project of Science and Technology Plan of Beijing Municipal Education Commission (KM202110005028), Project of Interdisciplinary Research Institute of Beijing University of Technology (2021020101) and International Research Cooperation Seed Fund of Beijing University of Technology (2021A01).

Table 8
Detailed results of multiple models on UCR datasets.

Datasets	ED	DTW	MSM	LS	FS	BOSS	TSF	TSBF	LPS	ST	EE	COTE	MLP	FCN	ResNet	OS-CNN	TSC-FF	EMAN	DMS-CNNs	AFFNet
Adiac	0.611	0.604	0.636	0.522	0.593	0.765	0.731	0.727	0.765	0.783	0.665	0.790	0.752	**0.857**	0.826	0.835	0.788	0.808	0.691	0.831
ArrowHead	0.800	0.703	0.815	0.846	0.594	0.834	0.726	0.801	0.806	0.737	0.811	0.811	0.823	**0.880**	0.817	0.838	0.803	0.846	0.851	0.800
Beef	0.667	0.633	0.474	0.867	0.567	0.800	0.767	0.554	0.520	**0.900**	0.633	0.867	0.833	0.750	0.767	0.807	0.844	0.700	0.733	0.833
BeetleFly	0.750	0.700	0.794	0.800	0.700	0.900	0.750	0.799	0.893	0.900	0.750	0.800	0.850	**0.950**	0.800	0.815	0.889	0.850	0.850	0.850
BirdChicken	0.550	0.750	0.866	0.800	0.750	**0.950**	0.800	0.902	0.854	0.800	0.800	0.900	0.800	**0.950**	0.900	0.885	0.922	0.900	0.917	0.900
Car	0.733	0.733	0.841	0.767	0.750	0.833	0.767	0.795	0.836	0.917	0.833	0.900	0.833	0.917	0.933	0.933	**0.950**	0.883	0.867	0.900
CBF	0.852	0.997	0.972	0.991	0.940	0.998	0.994	0.977	0.984	0.974	0.998	0.996	0.860	**1.000**	0.994	**1.000**	**1.000**	**1.000**	0.998	0.994
Chlorine	0.650	0.648	0.626	0.592	0.546	0.661	0.720	0.683	0.642	0.700	0.656	0.727	**0.872**	0.843	0.828	0.839	0.725	0.796	0.790	0.862
CinCECGTorso	0.897	0.651	0.935	0.870	0.859	0.887	0.983	0.716	0.743	0.954	0.942	**0.995**	0.842	0.813	0.771	0.827	0.951	0.664	0.743	0.761
Coffee	**1.000**	**1.000**	0.945	**1.000**	0.929	**1.000**	0.964	0.982	0.950	0.964	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
Computers	0.576	0.700	0.713	0.584	0.500	0.756	0.720	0.765	0.726	0.736	0.708	0.740	0.540	**0.848**	0.824	0.707	0.720	0.756	0.683	0.840
CricketX	0.577	0.754	0.778	0.741	0.485	0.736	0.664	0.731	0.696	0.772	0.813	0.808	0.569	0.815	0.821	**0.855**	0.808	0.805	0.762	0.844
CricketY	0.567	0.744	0.760	0.718	0.531	0.754	0.672	0.728	0.706	0.779	0.805	0.826	0.595	0.792	0.805	0.867	0.800	0.785	0.756	**0.869**
CricketZ	0.587	0.754	0.779	0.741	0.464	0.746	0.672	0.738	0.714	0.787	0.782	0.815	0.592	0.813	0.813	0.863	0.828	0.821	0.760	**0.869**
DiatomSizeR	0.935	0.967	0.939	0.980	0.866	0.931	0.931	0.890	0.915	0.925	0.944	0.928	0.964	0.930	0.931	0.977	0.947	0.964	**0.987**	0.941
DisPhxAgeGp	0.626	0.770	0.756	0.719	0.655	0.748	0.748	0.816	0.767	0.770	0.691	0.748	0.827	0.835	0.798	0.738	0.766	**0.860**	0.853	0.741
DisPhxCorr	0.717	0.717	0.754	0.779	0.750	0.728	0.772	0.812	0.742	0.775	0.728	0.761	0.810	0.812	0.820	0.766	0.712	**0.832**	0.802	0.764
DisPhxTW	0.633	0.590	0.618	0.626	0.626	0.676	0.669	0.690	0.618	0.662	0.647	0.698	0.747	0.790	0.740	0.664	0.688	**0.800**	0.798	0.698
Earthquakes	0.712	0.719	0.695	0.741	0.705	0.748	0.748	0.747	0.668	0.741	0.741	0.748	0.792	0.801	0.786	0.670	0.816	0.814	**0.820**	0.748
ECG200	0.880	0.770	0.877	0.880	0.810	0.870	0.870	0.847	0.807	0.830	0.880	0.880	**0.920**	0.900	0.870	0.908	0.892	**0.920**	0.883	**0.920**
ECG5000	0.925	0.924	0.930	0.932	0.923	0.941	0.939	0.938	0.917	0.944	0.939	0.946	0.935	0.941	0.931	0.940	**0.948**	0.946	0.944	0.941
ECGFiveDays	0.797	0.768	0.879	**1.000**	0.998	**1.000**	0.956	0.849	0.840	0.984	0.820	0.999	0.970	0.985	0.955	**1.000**	0.999	**1.000**	**1.000**	**1.000**
ElectricDevices	0.552	0.602	0.825	0.587	0.579	0.799	0.693	0.808	**0.853**	0.747	0.663	0.713	0.580	0.723	0.728	0.724	0.741	0.718	0.693	0.743
FaceAll	0.714	0.808	**0.986**	0.749	0.626	0.782	0.751	0.942	0.962	0.779	0.849	0.918	0.885	0.929	0.834	0.845	0.871	0.899	0.822	0.918
FaceFour	0.784	0.830	0.920	0.966	0.909	**1.000**	0.932	0.862	0.889	0.852	0.909	0.898	0.830	0.932	0.932	0.951	0.909	0.955	0.822	0.943
FacesUCR	0.769	0.905	0.970	0.939	0.706	0.957	0.883	0.849	0.910	0.906	0.945	0.942	0.815	0.948	0.958	0.967	0.959	0.945	0.896	**0.976**
FiftyWords	0.631	0.690	0.817	0.730	0.481	0.705	0.741	0.744	0.776	0.705	**0.820**	0.798	0.712	0.679	0.727	0.816	0.804	0.758	0.738	0.815
Fish	0.783	0.823	0.897	0.960	0.783	**0.989**	0.794	0.913	0.912	**0.989**	0.966	0.983	0.874	0.971	**0.989**	0.987	**0.989**	0.966	0.924	0.971
FordA	0.665	0.555	0.725	0.957	0.787	0.930	0.815	0.831	0.869	0.971	0.738	0.957	0.769	0.906	0.928	0.955	**1.000**	0.929	0.913	0.951
FordB	0.606	0.620	0.730	0.917	0.728	0.711	0.688	0.751	0.852	0.807	0.662	0.804	0.629	0.883	0.900	0.838	**1.000**	0.903	0.889	0.856
GunPoint	0.913	0.907	0.948	**1.000**	0.947	**1.000**	0.973	0.965	0.972	**1.000**	0.993	**1.000**	0.933	**1.000**	0.993	0.999	**1.000**	0.993	0.993	0.993
Ham	0.600	0.467	0.745	0.667	0.648	0.667	0.743	0.711	0.685	0.686	0.571	0.648	0.714	0.762	0.781	0.704	0.713	0.800	0.752	**0.829**
HandOutlines	0.862	0.881	0.864	0.481	0.811	0.903	0.919	0.879	0.868	**0.932**	0.889	0.919	0.807	0.776	0.861	0.929	0.907	0.886	0.885	0.897
Haptics	0.370	0.377	0.444	0.468	0.393	0.461	0.445	0.463	0.415	0.523	0.393	0.523	0.461	**0.551**	0.506	0.510	0.521	0.490	0.478	0.506
Herring	0.516	0.531	0.559	0.625	0.531	0.547	0.609	0.590	0.549	0.672	0.578	0.625	0.687	**0.703**	0.594	0.608	0.649	0.641	0.656	0.672
InlineSkate	0.342	0.384	0.455	0.438	0.189	**0.516**	0.376	0.377	0.449	0.373	0.460	0.495	0.351	0.411	0.365	0.429	0.462	0.442	0.427	0.336
InsectWing	0.562	0.355	0.570	0.606	0.489	0.523	0.633	0.616	0.519	0.627	0.595	**0.653**	0.631	0.402	0.531	0.635	0.627	0.645	0.638	0.652
ItalyPower	0.955	0.950	0.936	0.960	0.917	0.909	0.960	0.926	0.914	0.948	0.962	0.961	0.966	0.970	0.960	0.947	0.963	0.967	0.970	**0.975**
LrgKitApp	0.493	0.795	0.749	0.701	0.560	0.765	0.571	0.551	0.680	0.859	0.811	0.845	0.480	0.896	0.893	0.896	0.872	0.907	0.880	**0.917**
Lightning2	0.754	0.869	0.792	0.820	0.705	0.836	0.803	0.760	0.757	0.738	**0.885**	0.869	0.721	0.803	0.754	0.807	0.816	0.852	0.842	0.852
Lightning7	0.575	0.726	0.713	0.795	0.644	0.685	0.753	0.680	0.631	0.726	0.767	0.808	0.644	**0.863**	0.836	0.793	0.812	0.836	0.836	0.849
Mallat	0.914	0.934	0.918	0.950	0.976	0.938	0.919	0.951	0.908	0.964	0.940	0.954	0.936	**0.980**	0.979	0.964	0.954	0.967	0.948	0.958
Meat	0.933	0.933	0.977	0.733	0.833	0.900	0.933	0.983	0.968	0.850	0.933	0.917	0.933	0.967	**1.000**	0.947	0.933	0.950	0.944	0.950
MedicalImages	0.684	0.737	0.757	0.664	0.624	0.718	0.755	0.701	0.710	0.670	0.742	0.758	0.729	**0.792**	0.772	0.769	0.758	0.762	0.739	0.791
MidPhxAgeGp	0.519	0.500	0.751	0.571	0.545	0.545	0.578	**0.800**	0.770	0.643	0.558	0.636	0.735	0.768	0.760	0.536	0.700	0.795	**0.800**	0.545
MidPhxCorr	0.766	0.698	0.560	0.780	0.729	0.780	**0.828**	0.673	0.597	0.794	0.784	0.804	0.760	0.795	0.793	0.814	0.766	0.788	0.549	0.818
MidPhxTW	0.513	0.506	0.499	0.506	0.532	0.545	0.565	0.568	0.503	0.519	0.513	0.571	0.609	0.612	0.607	0.519	0.575	**0.649**	0.629	0.500
MoteStrain	0.879	0.835	0.880	0.883	0.777	0.879	0.869	0.886	0.917	0.897	0.883	0.937	0.869	**0.950**	0.895	0.926	0.894	0.890	0.869	0.899
NonInvThor1	0.829	0.790	0.818	0.259	0.710	0.838	0.876	0.842	0.807	0.950	0.846	0.931	0.942	0.961	0.948	**0.963**	0.882	0.952	0.945	0.952
NonInvThor2	0.880	0.865	0.894	0.770	0.754	0.901	0.910	0.862	0.826	0.951	0.913	0.946	0.943	0.955	0.951	**0.960**	0.942	0.947	0.949	0.952
OliveOil	0.867	0.833	0.872	0.167	0.733	0.867	0.867	0.864	0.892	**0.900**	0.867	**0.900**	0.400	0.833	0.867	0.787	0.867	0.867	0.422	0.767
OSULeaf	0.521	0.591	0.787	0.777	0.678	0.955	0.583	0.678	0.763	0.967	0.806	0.967	0.570	**0.988**	0.979	0.940	0.970	0.897	0.726	0.983
PhalCorr	0.761	0.728	0.760	0.765	0.744	0.772	0.803	0.825	0.790	0.763	0.773	0.770	**0.830**	0.826	0.825	**0.830**	0.810	0.817	0.809	0.815
Phoneme	0.109	0.228	0.275	0.218	0.174	0.265	0.212	0.278	0.245	0.321	0.305	0.349	0.098	0.345	0.324	0.305	0.306	0.245	0.153	**0.359**
Plane	0.962	**1.000**	0.999	**1.000**	**1.000**	**1.000**	**1.000**	0.993	**1.000**	**1.000**	**1.000**	**1.000**	0.981	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
ProxPhxAgeGp	0.785	0.805	0.806	0.834	0.780	0.834	0.849	0.861	0.851	0.844	0.805	0.854	0.824	0.849	0.849	0.844	0.854	**0.873**	0.854	0.844
ProxPhxCorr	0.808	0.784	0.769	0.849	0.804	0.849	0.828	0.842	0.800	0.883	0.808	0.869	0.887	0.900	**0.918**	0.908	0.876	0.890	0.893	0.918
ProxPhxTW	0.707	0.761	0.729	0.776	0.702	0.800	0.815	0.798	0.722	0.805	0.766	0.780	0.797	0.810	0.807	0.773	0.804	**0.825**	0.815	0.800
RefrigerationDevices	0.395	0.464	**0.704**	0.515	0.333	0.499	0.589	0.638	0.675	0.581	0.437	0.547	0.371	0.533	0.528	0.503	0.570	0.555	0.492	0.552
ScreenType	0.360	0.397	0.493	0.429	0.413	0.464	0.456	0.538	0.506	0.520	0.445	0.547	0.408	0.667	**0.707**	0.526	0.539	0.547	0.458	0.613
ShapeletSim	0.539	0.650	0.850	0.950	**1.000**	**1.000**	0.478	0.913	0.874	0.956	0.817	0.961	0.483	0.867	**1.000**	0.799	0.806	0.994	0.922	**1.000**
ShapesAll	0.752	0.768	0.875	0.768	0.580	0.908	0.792	0.853	0.885	0.842	0.867	0.892	0.775	0.898	0.912	0.920	0.815	0.883	0.842	**0.937**
SmlKitApp	0.344	0.643	0.717	0.664	0.333	0.725	0.811	0.674	0.724	0.792	0.696	0.776	0.389	0.803	0.797	0.721	**0.815**	0.787	0.673	0.795
SonyAIBORobot1	0.696	0.725	0.764	0.810	0.686	0.632	0.787	0.839	0.842	0.844	0.704	0.845	0.727	0.968	**0.985**	0.980	0.763	0.923	0.961	0.895
SonyAIBORobot2	0.859	0.831	0.877	0.875	0.790	0.859	0.810	0.825	0.851	0.934	0.878	0.952	0.839	0.962	0.962	0.954	0.902	0.919	0.887	**0.967**
StarLightCurves	0.849	0.907	0.882	0.947	0.918	0.978	0.969	0.978	0.968	0.979	0.926	**0.980**	0.957	0.967	0.975	0.975	**0.980**	0.968	0.975	0.978
Strawberry	0.946	0.941	0.958	0.911	0.903	0.976	0.965	0.968	0.963	0.962	0.946	0.951	0.967	0.969	0.958	**0.982**	0.971	0.974	0.969	0.976
SwedishLeaf	0.789	0.792	0.887	0.907	0.768	0.922	0.914	0.908	0.926	0.928	0.915	0.955	0.893	0.966	0.958	0.971	0.878	0.933	0.929	**0.976**
Symbols	0.899	0.950	0.952	0.932	0.934	0.967	0.915	0.944	0.960	0.882	0.960	0.964	0.853	0.962	0.872	0.961	0.800	0.956	0.938	**0.975**
SyntheticControl	0.880	0.993	0.982	0.997	0.910	0.967	0.987	0.987	0.972	0.983	0.990	**1.000**	0.950	0.990	**1.000**	0.999	**1.000**	0.997	**1.000**	**1.000**
ToeSegmentation1	0.680	0.772	0.821	0.934	0.956	0.939	0.741	0.858	0.841	0.965	0.829	0.974	0.601	0.969	0.965	0.954	**0.979**	0.961	0.934	0.974
ToeSegmentation2	0.808	0.838	0.895	0.915	0.692	**0.962**	0.815	0.886	0.926	0.908	0.892	0.915	0.746	0.915	0.862	0.946	0.932	0.908	0.931	0.923
Trace	0.760	**1.000**	0.956	**1.000**	**1.000**	**1.000**	0.990	0.981	0.966	**1.000**	0.990	**1.000**	0.820	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
TwoLeadECG	0.747	0.905	0.941	0.996	0.924	0.981	0.759	0.910	0.928	0.997	0.971	0.993	0.886	0.897	**1.000**	0.999	0.929	0.993	0.997	0.998
TwoPatterns	0.907	**1.000**	0.999	0.993	0.908	0.993	0.991	0.974	0.967	0.955	**1.000**	**1.000**	0.853	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	0.999	**1.000**
UWaveGestAll	0.948	0.892	0.775	0.953	0.789	0.939	0.957	0.834	0.819	0.942	0.968	0.964	0.954	0.826	0.868	0.942	**0.971**	0.966	0.946	0.951
UWaveGestX	0.739	0.728	0.690	0.791	0.695	0.762	0.804	0.746	0.753	0.803	0.805	0.822	0.768	0.754	0.787	0.822	0.846	0.812	0.791	**0.853**
UWaveGestY	0.662	0.634	0.701	0.703	0.596	0.685	0.727	0.776	0.766	0.730	0.726	0.759	0.703	0.725	0.668	0.757	**0.784**	0.733	0.706	0.776
UWaveGestZ	0.650	0.658	0.960	0.747	0.638	0.695	0.743	0.944	**0.968**	0.748	0.724	0.750	0.705	0.729	0.755	0.764	0.792	0.760	0.742	0.791
Wafer	0.995	0.980	0.996	0.996	0.997	0.995	0.996	0.996	0.995	**1.000**	0.997	**1.000**	0.996	0.997	0.997	0.998	**1.000**	0.999	0.998	0.999
Wine	0.611	0.574	0.884	0.500	0.759	0.741	0.630	0.879	0.884	0.796	0.574	0.648	0.796	**0.889**	0.796	0.744	0.633	0.796	0.556	0.722
WordSynonyms	0.618	0.649	0.773	0.607	0.431	0.638	0.647	0.669	0.728	0.571	**0.779**	0.757	0.594	0.580	0.632	0.742	0.661	0.674	0.654	0.726
Worms	0.455	0.584	0.616	0.610	0.649	0.558	0.610	0.668	0.642	0.740	0.662	0.623	0.343	0.669	0.619	0.765	0.652	0.630	0.552	**0.831**
WormsTwoClass	0.610	0.623	0.712	0.727	0.727	0.831	0.623	0.755	0.743	0.831	0.688	0.805	0.597	0.729	0.735	0.657	0.795	0.757	0.766	**0.857**
Yoga	0.830	0.837	0.888	0.834	0.695	**0.918**	0.859	0.835	0.874	0.818	0.879	0.877	0.855	0.845	0.858	0.911	0.901	0.874	0.850	0.879

References

[1] L. Anghinoni, L. Zhao, D. Ji, H. Pan, Time series trend detection and forecasting using complex network topology analysis, Neural Netw. 117 (2019) 295–306.

[2] Y. Gao, S.S. Vedula, C.E. Reiley, N. Ahmidi, B. Varadarajan, H.C. Lin, L. Tao, L. Zappella, B. Béjar, D.D. Yuh, et al., Jhu-isi gesture and skill assessment working set (jigsaws): A surgical activity dataset for human motion modeling, in: MICCAI Workshop: M2cai, Vol. 3, 2014, p. 3.

[3] X. Yuan, L. Li, Y.A. Shardt, Y. Wang, C. Yang, Deep learning with spatiotemporal attention-based lstm for industrial soft sensor model development, IEEE Trans. Ind. Electron. 68 (5) (2020) 4404–4414.

[4] M. Längkvist, L. Karlsson, A. Loutfi, A review of unsupervised feature learning and deep learning for time-series modeling, Pattern Recognit. Lett. 42 (2014) 11–24.

[5] D.J. Berndt, J. Clifford, Using dynamic time warping to find patterns in time series, in: KDD Workshop, Vol. 10, Seattle, WA, USA, 1994, pp. 359–370.

[6] H. Deng, G. Runger, E. Tuv, M. Vladimir, A time series forest for classification and feature extraction, Inform. Sci. 239 (2013) 142–153.

[7] L. Ye, E. Keogh, Time series shapelets: a new primitive for data mining, in: Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2009, pp. 947–956.

[8] P. Senin, S. Malinchik, Sax-vsm: Interpretable time series classification using sax and vector space model, in: 2013 IEEE 13th International Conference on Data Mining, IEEE, 2013, pp. 1175–1180.

[9] J. Lines, S. Taylor, A. Bagnall, Time series classification with hive-cote: The hierarchical vote collective of transformation-based ensembles, ACM Trans. Knowl. Discov. Data 12 (5) (2018).

[10] A. Shifaz, C. Pelletier, F. Petitjean, G.I. Webb, Ts-chief: a scalable and accurate forest algorithm for time series classification, Data Min. Knowl. Discov. 34 (3) (2020) 742–775.

[11] M. Waqas, S. Tu, Z. Halim, S.U. Rehman, G. Abbas, Z.H. Abbas, The role of artificial intelligence and machine learning in wireless networks security: principle, practice and challenges, Artif. Intell. Rev. (2022) 1–47.

[12] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 770–778.

[13] T.B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, et al., Language models are few-shot learners, 2020, arXiv preprint arXiv:2005.14165.

[14] T. Karras, S. Laine, T. Aila, A style-based generator architecture for generative adversarial networks, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 4401–4410.

[15] H.I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, P.-A. Muller, Deep learning for time series classification: a review, Data Min. Knowl. Discov. 33 (4) (2019) 917–963.

[16] Y. Zheng, Q. Liu, E. Chen, Y. Ge, J.L. Zhao, Exploiting multi-channels deep convolutional neural networks for multivariate time series classification, Front. Comput. Sci. 10 (1) (2016) 96–112.

[17] Z. Cui, W. Chen, Y. Chen, Multi-scale convolutional neural networks for time series classification, 2016, arXiv preprint arXiv:1603.06995.

[18] W. Tang, G. Long, L. Liu, T. Zhou, J. Jiang, M. Blumenstein, Rethinking 1d-cnn for time series classification: A stronger baseline, 2020, arXiv preprint arXiv:2002.10061.

[19] B.K. Iwana, S. Uchida, Time series classification using local distance-based features in multi-modal fusion networks, Pattern Recognit. 97 (2020) 107024.

[20] B. Wang, T. Jiang, X. Zhou, B. Ma, F. Zhao, Y. Wang, Time-series classification based on fusion features of sequence and visualization, Appl. Sci. 10 (12) (2020) 4124.

[21] A. Abanda, U. Mori, J.A. Lozano, A review on distance based time series classification, Data Min. Knowl. Discov. 33 (2) (2019) 378–412.

[22] S. Tu, M. Waqas, S.U. Rehman, T. Mir, G. Abbas, Z.H. Abbas, Z. Halim, I. Ahmad, Reinforcement learning assisted impersonation attack detection in device-to-device communications, IEEE Trans. Veh. Technol. 70 (2) (2021) 1474–1479.

[23] P. Schäfer, The boss is concerned with time series classification in the presence of noise, Data Min. Knowl. Discov. 29 (6) (2015) 1505–1530.

[24] A. Dempster, F. Petitjean, G.I. Webb, Rocket: exceptionally fast and accurate time series classification using random convolutional kernels, Data Min. Knowl. Discov. 34 (5) (2020) 1454–1495.

[25] T. Rakthanmanon, E. Keogh, Fast shapelets: A scalable algorithm for discovering time series shapelets, in: Proceedings of the 2013 SIAM International Conference on Data Mining, SIAM, 2013, pp. 668–676.

[26] J. Grabocka, N. Schilling, M. Wistuba, L. Schmidt-Thieme, Learning time-series shapelets, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 392–401.

[27] J. Lines, A. Bagnall, Time series classification with ensembles of elastic distance measures, Data Min. Knowl. Discov. 29 (3) (2015) 565–592.

[28] A. Bagnall, J. Lines, J. Hills, A. Bostrom, Time-series classification with cote: the collective of transformation-based ensembles, IEEE Trans. Knowl. Data Eng. 27 (9) (2015) 2522–2535.

[29] K. He, G. Gkioxari, P. Dollár, R. Girshick, Mask r-cnn, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 2961–2969.

[30] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A.N. Gomez, Ł. Kaiser, I. Polosukhin, Attention is all you need, in: Advances in Neural Information Processing Systems, 2017, pp. 5998–6008.

[31] Z. Wang, W. Yan, T. Oates, Time series classification from scratch with deep neural networks: A strong baseline, in: 2017 International Joint Conference on Neural Networks (IJCNN), IEEE, 2017, pp. 1578–1585.

[32] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, G. Shroff, Convtimenet: A pre-trained deep convolutional neural network for time series classification, in: 2019 International Joint Conference on Neural Networks (IJCNN), IEEE, 2019, pp. 1–8.

[33] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, A. Rabinovich, Going deeper with convolutions, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2015, pp. 1–9.

[34] H.I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D.F. Schmidt, J. Weber, G.I. Webb, L. Idoumghar, P.-A. Muller, F. Petitjean, Inceptiontime: Finding alexnet for time series classification, Data Min. Knowl. Discov. 34 (6) (2020) 1936–1962.

[35] X. Zhang, Y. Gao, J. Lin, C.-T. Lu, Tapnet: Multivariate time series classification with attentional prototypical network, in: Proceedings of the AAAI Conference on Artificial Intelligence Vol. 34, 2020, pp. 6845–6852.

[36] Q. Ma, Z. Zheng, W. Zhuang, E. Chen, J. Wei, J. Wang, Echo memory-augmented network for time series classification, Neural Netw. 133 (2021) 177–192.

[37] H. Jaeger, H. Haas, Harnessing nonlinearity: Predicting chaotic systems and saving energy in wireless communication, Science 304 (5667) (2004) 78–80.

[38] Y. Chen, X. Dai, M. Liu, D. Chen, L. Yuan, Z. Liu, Dynamic convolution: Attention over convolution kernels, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2020, pp. 11030–11039.

[39] B. Yang, G. Bender, Q.V. Le, J. Ngiam, CondConv: conditionally parameterized convolutions for efficient inference, in: Proceedings of the 33rd International Conference on Neural Information Processing Systems, 2019, pp. 1307–1318.

[40] J. Hu, L. Shen, G. Sun, Squeeze-and-excitation networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7132–7141.

[41] Y. Zhang, J. Zhang, Q. Wang, Z. Zhong, Dynet: Dynamic convolution for accelerating convolutional neural networks, 2020, arXiv preprint arXiv:2004.10694.

[42] N. Ma, X. Zhang, J. Huang, J. Sun, Weightnet: Revisiting the design space of weight networks, in: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part XV 16, Springer, 2020, pp. 776–792.

[43] Y. Li, Y. Chen, X. Dai, M. Liu, D. Chen, Y. Yu, L. Yuan, Z. Liu, M. Chen, N. Vasconcelos, Revisiting dynamic convolution via matrix decomposition, 2021, arXiv preprint arXiv:2103.08756.

[44] B. Qian, Y. Xiao, Z. Zheng, M. Zhou, W. Zhuang, S. Li, Q. Ma, Dynamic multi-scale convolutional neural network for time series classification, IEEE Access 8 (2020) 109732–109746.

[45] T.-Y. Lin, P. Dollár, R. Girshick, K. He, B. Hariharan, S. Belongie, Feature pyramid networks for object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2117–2125.

[46] G. Huang, Z. Liu, L. Van Der Maaten, K.Q. Weinberger, Densely connected convolutional networks, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 4700–4708.

[47] Y. Dai, F. Gieseke, S. Oehmcke, Y. Wu, K. Barnard, Attentional feature fusion, in: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, 2021, pp. 3560–3569.

[48] W. Zhou, H. Li, J. Sun, Q. Tian, Collaborative index embedding for image retrieval, IEEE Trans. Pattern Anal. Mach. Intell. 40 (5) (2017) 1154–1166.

[49] H. Zhang, M. Cisse, Y.N. Dauphin, D. Lopez-Paz, Mixup: Beyond empirical risk minimization, in: International Conference on Learning Representations, 2018.

[50] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens, Z. Wojna, Rethinking the inception architecture for computer vision, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 2818–2826.

[51] H.A. Dau, A. Bagnall, K. Kamgar, C.-C.M. Yeh, Y. Zhu, S. Gharghabi, C.A. Ratanamahatana, E. Keogh, The ucr time series archive, IEEE/CAA J. Autom. Sin. 6 (6) (2019) 1293–1305.

[52] J.R. Kwapisz, G.M. Weiss, S.A. Moore, Activity recognition using cell phone accelerometers, ACM SigKDD Explor. Newsl. 12 (2) (2011) 74–82.

[53] D. Anguita, A. Ghio, L. Oneto, X. Parra, J.L. Reyes-Ortiz, et al., A public domain dataset for human activity recognition using smartphones., in: Esann, Vol. 3, 2013, p. 3.

[54] H. Sagha, S.T. Digumarti, J.d.R. Millán, R. Chavarriaga, A. Calatroni, D. Roggen, G. Tröster, Benchmarking classification techniques using the opportunity human activity dataset, in: 2011 IEEE International Conference on Systems, Man, and Cybernetics, IEEE, 2011, pp. 36–40.

[55] A. Ignatov, Real-time human activity recognition from accelerometer data using convolutional neural networks, Appl. Soft Comput. 62 (2018) 915–922.

[56] Y. Zhao, R. Yang, G. Chevalier, X. Xu, Z. Zhang, Deep residual bidir-LSTM for human activity recognition using wearable sensors, Math. Probl. Eng. 2018 (2018).

[57] D.P. Kingma, J. Ba, Adam: A method for stochastic optimization, 2014, arXiv preprint arXiv:1412.6980.

[58] A. Bagnall, J. Lines, A. Bostrom, J. Large, E. Keogh, The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances, Data Min. Knowl. Discov. 31 (3) (2017) 606–660.

[59] A. Stefan, V. Athitsos, G. Das, The move-split-merge metric for time series, IEEE Trans. Knowl. Data Eng. 25 (6) (2012) 1425–1438.

[60] M.G. Baydogan, G. Runger, E. Tuv, A bag-of-features framework to classify time series, IEEE Trans. Pattern Anal. Mach. Intell. 35 (11) (2013) 2796–2802.

[61] M.G. Baydogan, G. Runger, Time series representation and similarity based on local autopatterns, Data Min. Knowl. Discov. 30 (2) (2016) 476–509.

[62] J. Hills, J. Lines, E. Baranauskas, J. Mapp, A. Bagnall, Classification of time series by shapelet transformation, Data Min. Knowl. Discov. 28 (4) (2014) 851–881.

[63] B. Kolosnjaji, C. Eckert, Neural network-based user-independent physical activity recognition for mobile devices, in: International Conference on Intelligent Data Engineering and Automated Learning, Springer, 2015, pp. 378–386.

[64] C.A. Ronao, S.-B. Cho, Human activity recognition with smartphone sensors using deep learning neural networks, Expert Syst. Appl. 59 (2016) 235–244.

[65] J. Yang, M.N. Nguyen, P.P. San, X.L. Li, S. Krishnaswamy, Deep convolutional neural networks on multichannel time series for human activity recognition, in: Twenty-Fourth International Joint Conference on Artificial Intelligence, 2015.

[66] S. Tu, M. Waqas, S.U. Rehman, T. Mir, Z. Halim, I. Ahmad, Social phenomena and fog computing networks: A novel perspective for future networks, IEEE Trans. Comput. Soc. Syst. (2021).