A New Attention Mechanism to Classify Multivariate Time Series

Yifan Hao and **Huiping Cao**
New Mexico State University
{yifan, hcao}@nmsu.edu

Abstract

Classifying multivariate time series (MTS), which record the values of multiple variables over a continuous period of time, has gained a lot of attention. However, existing techniques suffer from two major issues. First, the long-range dependencies of the time-series sequences are not well captured. Second, the interactions of multiple variables are generally not represented in features. To address these aforementioned issues, we propose a novel Cross Attention Stabilized Fully Convolutional Neural Network (CA-SFCN) to classify MTS data. First, we introduce a temporal attention mechanism to extract long- and short-term memories across all time steps. Second, variable attention is designed to select relevant variables at each time step. CA-SFCN is compared with 16 approaches using 14 different MTS datasets. The extensive experimental results show that the CA-SFCN outperforms state-of-the-art classification methods, and the cross attention mechanism achieves better performance than other attention mechanisms.

1 Introduction

Classifying multivariate time series (MTS), which record the values of multiple variables over a continuous period of time, has recently gained a lot of attention [Yang *et al.*, 2015; Karim *et al.*, 2018; Karim *et al.*, 2019]. In analyzing time series data, it is important to utilize the dependencies of values in the series. A value that happened at a time step t may depend on its immediate historical values or the historical values that happened far before t. Such dependencies are denoted as *short-term* and *long-term* dependencies respectively.

For MTS data, which contains multiple variables, the analysis needs to further consider the relationships among these variables. Let us use an example to illustrate the need. Given two words in the Australian sign language: "suffer"[i] and "arithmetic"[ii]. If we consider either the left hand or the right hand or both hands (without synchronizing two hands) together using Dynamic Warping distance (which captures the shape similarity), the hand movement for these two words is similar. However, if we consider the two hands together and their movement dependency, the hand movement for these words is different: "arithmetic" is represented by moving both hands towards the same direction (upward or downward), while the word "suffer" is expressed by moving the two hands in opposite directions (one moves upward and the other moves downward). The two words can be differentiated when we consider the two hands' interaction.

Most recent models of classifying MTS data are Convolutional Neural Networks (CNN) or Recurrent Neural Networks (RNN) based. These models have two major limitations. First, these models can not effectively capture and utilize long-term dependencies. Second, the interactions of multiple variables are not well studied.

CNN-based models generate features by combining local neighborhood information. The long-term dependencies cannot be captured in shallow CNN networks. RNN-based models calculate the features at a time step by using the hidden states at the previous time step. The long-term memories can be captured by late CNN layers in deep CNN models or RNN-based methods. However, experiments show that both CNN and RNN based approaches still have problems with capturing dependencies to *very long* historical data. Furthermore, it is computationally very inefficient to train such deep neural networks [Zhang *et al.*, 2019].

Traditional feature extraction methods, such as principal component analysis [Yoon *et al.*, 2005], Random forest [Ho, 1995], support vector machine [Boser *et al.*, 1992], treat the time series data as order-free vectors. Experiments show that simply applying those techniques on MTS misses information [Hao *et al.*, 2019].

Another everlasting challenge in the design of neural networks (in particular deep network) models is their efficient training. More recently, CNN-based models gain more interest than RNN-based models. RNN-based models generally require sequential loops in its training because, to calculate the hidden state at each time step t, the model first needs to calculate the hidden state before t. Such sequential loops cannot be easily parallelized to improve the training efficiency. However, CNN-based models can be better trained in parallel and accelerated using GPU.

To address these aforementioned two limitations and the computational challenges on MTS classification, we propose

[i] http://www.auslan.org.au/dictionary/words/suffer-1.html
[ii] http://www.auslan.org.au/dictionary/words/arithmetic-1.html

a novel Cross-Attention (CA) based Stabilized Fully Convolutional Network (CA-SFCN) to better classify MTS data. The design of the new cross attention is inspired by the attention mechanism, which was introduced in [Bahdanau *et al.*, 2015] to encode dependencies between two sequences in Natural Language Processing (NLP). CA consists of temporal attention (TA) and variable Attention (VA) to capture the long- and short-term memories in time series, and the interaction of multiple variables. Our newly designed CA is very different from other attention mechanisms, which is explained in the next section.

Multivariate time series (MTS) data records values for multiple variables. One MTS instance is denoted as $(v_1, v_2, \cdots, v_V)$, where $v_i = (v_i^1, v_i^2, \cdots, v_i^m)$ is one time series for the i-th variable, V is the number of variables and m is the time series length. Each MTS instance has one corresponding label. Later on, we will use superscript ^{t} to represent a time step. Our **research problem** is to accurately predict the correct class label of an MTS.

The paper is organized as follows. Section 2 discusses the literature. Section 3 explains our proposed approaches. Section 4 presents our experiments and shows the effectiveness and efficiency of our proposed approaches. Finally, Section 5 concludes our work.

2 Related Works

Time series classification has been extensively studied in the literature. Most of these methods (e.g., [Baydogan and Runger, 2015; Schäfer and Leser, 2017; Pei *et al.*, 2018; Tuncel and Baydogan, 2018]) extract features from the time series and utilize these features to classify the instances.

Recently, neural network-based approaches have achieved great success in MTS classification. In particular, CNN models can be used to capture short-term dependencies because convolutional operators in CNN process the information in a neighborhood (e.g., continuous sub-sequences in a time series). Deep CNN models can also capture long-term dependencies, but not effectively [Zhang *et al.*, 2019]. RNN models can represent the temporal dependencies in time series [Pascanu *et al.*, 2014]. We have seen successful stories of using such models to classify time series data [Yang *et al.*, 2015; Karim *et al.*, 2018].

More recent works use attention mechanisms because the attention idea represents the human intuition that some portion of data is given more emphasis when we look at a large dataset. Long Short-Term Memory Fully Convolutional Networks (LSTM-FCN) and Attention LSTM-FCN (ALSTM-FCN) [Karim *et al.*, 2018] have shown to be successful to classify uni-variate time series. They are recently adapted to classify MTS, denoted as Multivariate LSTM-FCN (MLSTM-FCN) and Multivariate Attention LSTM-FCN (MALSTM-FCN) [Karim *et al.*, 2019]. These two models can learn combined features by applying convolution operations on all variables [Karim *et al.*, 2019]. However, the combined features do not consider the pair-wise dependencies between two variables. When the variable number is large, it is hard to emphasize the interactions of a small group of variables. Also, these models cannot efficiently and effectively capture the long-term dependencies.

A recent Global Attention (GA) strategy is presented in [Zhang *et al.*, 2019] to extract the long-range dependencies from convolutional features of images. GA calculation ignores the order of the convolutional features in both image axes. The temporal dependencies can be captured by applying attention mechanisms on the hidden states of an RNN-based model [Qin *et al.*, 2017], which predicts the future values based on the historical values of a series. We call this mechanism Recurrent Attention (RA) in this paper. All the existing attention mechanisms can not be to directly applied to MTS data for their classification.

3 Our Approach: New Cross Attention (CA) Enabled Stabilized Fully Convolutional Networks (SFCN)

In this section, we introduce a new attention mechanism, cross attention (CA), and present a **CA-based Stabilized Fully Convolutional Networks (CA-SFCN)** to efficiently model historical (including long-term) dependencies and the complex interactions of the multiple variables.

Figure 1 shows the architecture of the newly proposed CA-SFCN. This architecture consists of (i) a component of several fully connected convolutional layers, (ii) a newly designed cross attention (CA) block, and (iii) a global pooling layer. The use of the fully connected convolutional layers are inspired by the design of Fully Convolutional Networks (FCN). FCN is first introduced in [Long *et al.*, 2015] for effective semantic segmentation of images. It is further used as a feature extractor in MTS classification [Wang *et al.*, 2017]. Different from traditional Convolutional Neural Networks (CNNs), FCN uses a global pooling layer to replace the fully connected layers before the output. FCN does not include a pooling layer after each convolutional layer. FCN has shown superior performance on MTS classification [Wang *et al.*, 2017]. To implement the fully convolutional networks, we use 2D convolutional filters. This is different from other FCN approaches (e.g.,[Karim *et al.*, 2019]), which use 1D convolutional filters. Using 1D convolutional filters combine all the variables at the first convolutional layer. The pair-wise dependencies among different variables can not be captured anymore since all variables are combined into one feature space.

The CA block includes two major modules to implement our cross attention mechanism, temporal attention (TA) module and variable attention (VA) module. The CA block first runs the TA module. TA module uses the output of the last convolutional layer $\mathbf{X}$ to calculate the features O_{TA} that leverage temporal attention. Then, O_{TA} is combined with the $\mathbf{X}$ again to get hidden states $\mathbf{Y}$ (Eq. (1)).

$$\mathbf{Y} = \gamma \cdot O_{TA} + \mathbf{X}, \text{ where } \gamma \text{ is a scalar value} \quad (1)$$

The VA module uses $\mathbf{Y}$ as input to calculate the features that accommodate the variable attention O_{VA}. O_{VA} is then combined with $\mathbf{Y}$ and get hidden states $\mathbf{Z}$ using Eq. (2).

$$\mathbf{Z} = \zeta \cdot O_{VA} + \mathbf{Y}, \text{ where } \zeta \text{ is a scalar value} \quad (2)$$

The final features in $\mathbf{Z}$ combining both the O_{TA} and O_{VA} are called cross attention features.

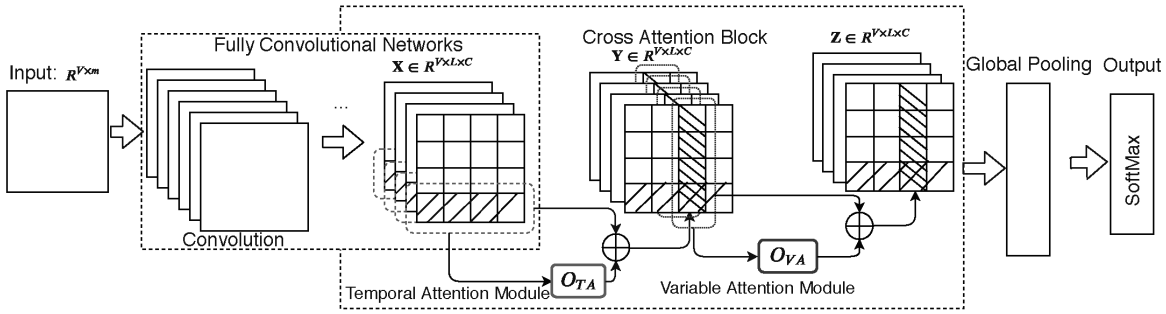


Figure 1: The architecture of CA-SFCN (Cross-Attention based Stabilized Fully Convolutional Networks)

Section 3.1 explains the detailed design of the new CA mechanism. Section 3.2 introduces the techniques to make the FCN more stabilized, which leads to the term Stabilized FCN (SFCN).

3.1 Cross Attention Mechanism

Cross Attention (CA) contains two attention modules: a temporal attention module that generates temporal attention (TA) and a variable attention module to generate variable attention (VA). TA captures the dependencies of the historical values in each time series. VA represents the variable attention module, which evaluates the dependencies of different variables.

Temporal Attention (TA) Calculation for the Time Series of One Variable

The goal of the TA module is to calculate TA for each variable. The TA of one variable captures the critical historical long-term and short-term dependencies for all the values in this variable's corresponding time series.

For one variable, let us use $X = (x_1^1, x_1^2, \ldots, x_1^L) \cdots (x_C^1, x_C^2, \ldots, x_C^L)$ to represent its feature sequences in the input layer or a convolutional layer in Figure 1. Here, L is the number of features for L time steps and C is the number of channels. For a special case that X is the data of the input layer, then $L = m$ and $C = 1$.

Fig. 2 shows the structure of the TA module. It demonstrates how to use the matrix $X \in R^{L \times C}$ to calculate the TA matrix $\alpha \in R^{L \times L}$ and the output of the TA module O_{TA}.

As the first step, X is first transformed to three feature spaces (Q, K, and V) using Eq. (3).

$$Q(X) = X \cdot W_Q, \;\; K(X) = X \cdot W_K, \;\; V(X) = X \cdot W_V \quad (3)$$

where $W_Q, W_K \in R^{C \times C_a}$ and $W_V \in R^{C \times C_v}$. The channel numbers for $Q(X)$ and $K(X)$, C_a, need to be the same. The number of channels for $V(X)$, C_v, can be different from C_a. Here, $Q(X) \in R^{L \times C_a}$ is generally called the query space, $K(X) \in R^{L \times C_a}$ is denoted as the key space, and $V(X) \in R^{L \times C_v}$ is called the value space. The three feature spaces are named following the convention of [Vaswani *et al.*, 2017] where the attention captures the mapping relationships between a possible query and the key-value pairs in the data. The transformation of X to each feature space is equivalent to and can be implemented as applying a 1×1 convolutional operation to X.

In the second step, the temporal attention for one variable, denoted as α in Figure 2, is calculated using the features in the query space and key space by two operations shown in Eq. (4) and Eq. (5).

$$S = Q(X) \cdot K(X)^T \quad (4)$$

Let $S_{q,k}$ be a hidden state in the matrix $S (\in R^{L \times L})$. The $S_{q,k}$ value captures the attention that the historical memory at time step k gives to the feature at time step q. Thus, $S_{q,k}$ is valid only when $k \leq q$. To accommodate this, S is updated to S' by setting $S_{i,j}$ to be zero when $i < j$. I.e., the right upper corner of the S matrix is set to zero. S directly captures the attention of both long-term and short-term historical values. For example, consider a sequence of length $L = 100$ and its last feature, $S_{L,L-99}$ is the attention that its last feature gets from the first time step, which can be treated as long-term dependency, while $S_{L,L-1}$ captures the attention that L's immediate previous step gives to the last feature, which is an example of short-term dependency.

The softmax function is applied to S' to normalize the attention as shown in Eq. (5).

$$\alpha_{q,k} = \frac{exp(S_{q,k})}{\sum_{j=1}^{q} exp(S_{q,j})} \;\; (1 \leq k \leq q \leq L) \quad (5)$$

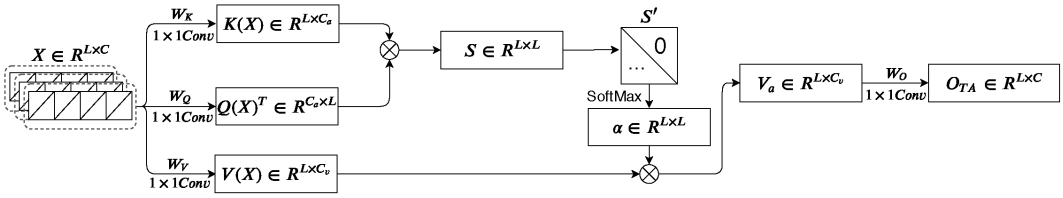
In the third step, the normalized attention α is applied to the features in the value space $V(X)$ to calculate the output of the attention, O_{TA}. This step first gets the attention hidden states $V_a = \alpha \cdot V(X)$, then calculates the output using Eq. (6).

$$O_{TA} = V_a \cdot W_o \;\; \text{where } W_o \in R^{C_v \times C} \quad (6)$$

The O_{TA} calculation can be implemented as applying a 1×1 convolutional operation to V_a.

The TA calculation is more efficient than typical RNN model training. As analyzed above, the training of RNN models generally requires sequential loops. However, in the TA calculation, only matrix multiplications are needed. Matrix multiplication can be easily calculated in parallel using GPU.

The temporal attention calculation is different from other existing attention mechanisms. As far as we know, existing attention mechanisms either do not consider the temporal order of the values [Zhang *et al.*, 2019] or calculates the attention from features that already represent the temporal order



$\otimes$ denotes matrix multiplication

Figure 2: Temporal Attention module to calculate the attention of one variable

of values (e.g., using recurrent layers) [Qin *et al.*, 2017]. The TA mechanism directly captures the time dependency of values in the attention calculation.

Variable Attention (CA) Calculation for Multiple Variables

Besides designing the aforementioned TA module to capture the long-term and short-term temporal dependency of values in one time-series sequence, we also design a Variable Attention (VA) module to evaluate the interactions between multiple variables.

The input sequence of VA is the feature sequence of all variables at the same time step from C channels, $Y^t = ((y^t_{1,1}, y^t_{2,1}, \cdots, y^t_{V,1}), \cdots, (y^t_{1,C}, y^t_{2,C}, \cdots, y^t_{V,C}))$. Using the similar procedure to calculate α in the TA module, we can calculate a normalized variable attention (say $\beta \in R^{V \times V}$). The output of applying the variable attention to Y^t is O_{VA} ($\in R^{V \times C}$).

Different from the TA calculation, which uses a horizontal slice of the feature tensor $\mathbf{X}$, VA calculation utilizes all the variables at the same time step. As shown in Fig. 1, Y^t is a vertical slice of the feature tensor $\mathbf{Y}$.

3.2 Techniques to Stabilize the FCN Training

The CA-based FCN is trained using a gradient descent algorithm through multiple iterations. In each iteration, a batch of instances is used to adjust the weights of the model. We introduce techniques to stabilize the model training by addressing two issues to better adjust the batch instances.

The first issue is with the batch normalization (BN) step, which is commonly used in training neural network models [Ioffe and Szegedy, 2015]. The exiting approach of BN is applied to all the instances in a batch. This, however, cannot represent the real situation that instances from different classes may have different distributions. To improve this step, we apply normalization on instances from different classes using a similar idea as that in [Miyato *et al.*, 2018].

The second issue is with the choice of instances in each batch. Regular choice of batch instances does not consider the situation of imbalanced instances in a batch, which however is common in the real situation. To address this issue, we define a minimal number of instances σ_{min} for each class. Then, in choosing the instance batches, the algorithm makes sure to sample at least σ_{min} instances from each class in every batch. A similar strategy is utilized in [Hao *et al.*, 2019].

The aforementioned BN and batch instance selection techniques are used to stabilize the training procedure. The stabilized FCN model is denoted as **SFCN** in this paper.

Dataset	N	CL	V	m
Action	560	20	570	100
Activity	320	16	570	337
Ara Voice	8800	88	39	91
Auslan	2565	95	22	96
Daliy Sport	9120	19	45	125
Ges	396	5	18	214
Har	10299	6	9	128
Ht Sensor	100	3	11	5396
JVowels	640	9	12	26
OHC	2858	20	30	173
Net	1337	2	4	994
Eeg	128	2	13	117
Eeg2	1200	2	64	256
Ozone	346	2	72	291

N: # of instances, CL: # of class labels, V: # of variables, and m: temporal steps of each time series

Table 1: Datasets

4 Experiments

All the methods are implemented using Python 3.7, and tested on a server with Intel Xeon Gold 5117 2.0G CPUs, 192GB RAM, and one Nvidia Tesla P100 GPU. TensorFlow (www.tensorflow.org) is used to build the CA-SFCN model and Adamoptimizer is used in the training process.

4.1 Methods for Comparison

The performance of the proposed approaches is compared with 16 approaches including (i) four existing state-of-the-art MTS classification approaches, (ii) nine other baseline methods, and (iii) SFCN without cross attention and SFCN with other attention mechanisms.

MTS Classification Approaches

The state-of-the-art approaches that we compare with include Multivariate Long Short Term Memory Fully Convolutional Network (MLSTM-FCN) and Attention MLSTM-FCN (AMLSTM-FCN in short) [Karim *et al.*, 2019] which extend the LSTM-FCN and ALTSM-FCN [Karim *et al.*, 2018] methods. They achieve the best performance on classifying MTS data by combining Long Short-Term Memory (LSTM) and FCN. Besides the previous four state-of-the-art approaches, 9 other baseline methods (denoted as OB), including WMUSE [Schäfer and Leser, 2017], ARKernel [Tuncel and Baydogan, 2018], LPS [Baydogan and Runger, 2016], mv-ARF [Tuncel and Baydogan, 2018], SMTS [Baydogan and Runger, 2015], HULM [Pei *et al.*, 2018], DTW [Seto *et*

Dataset	Methods								
	LSTM-FCN	MLSTM-FCN	ALSTM-FCN	MALSTM-FCN	Best-of-OB	GA-SFCN	RA-SFCN	SFCN	CA-SFCN
Action	0.717	0.754	0.727	0.747	0.707	0.810	0.819	0.808	**0.835**
Activity	0.531	0.619	0.556	0.588	0.581	0.610	0.607	0.606	**0.623**
Eeg	0.609	**0.656**	0.641	0.641	0.625	0.547	0.549	0.547	**0.656**
Eeg2	0.907	0.910	0.907	0.913	0.775	0.977	0.965	0.978	**0.983**
Ges	0.505	0.535	0.525	0.531	0.409	0.585	0.571	0.561	**0.591**
HT Sensor	0.680	0.780	0.720	**0.800**	0.720	**0.800**	**0.800**	**0.800**	**0.800**
Ozone	0.676	**0.815**	0.792	0.798	0.751	0.809	0.803	0.786	0.792
Ara Voice	0.980	0.980	**0.986**	0.983	0.946	0.972	0.965	0.965	0.980
Daily Sport	**0.997**	**0.997**	**0.997**	**0.997**	0.984	0.995	0.993	0.995	0.995
Net	0.940	0.950	0.930	0.950	**0.980**	0.953	0.949	0.943	0.951
Har	0.960	**0.967**	0.955	**0.967**	0.816	0.963	0.965	0.965	**0.967**
Auslan	0.970	0.970	0.960	0.960	**0.980**	0.977	0.970	0.977	0.978
JVowels	0.990	**1.000**	0.990	0.990	0.980	0.984	0.965	0.986	0.990
OHC	**1.000**	**1.000**	**1.000**	**1.000**	0.990	**1.000**	**1.000**	**1.000**	**1.000**

*Best-of-OB: the best results from all the other 9 baseline approaches

Table 2: Classification performance comparison (the results of our newly proposed method *CA-SFCN* are shown in the last column)

al., 2015], SVM [Boser *et al.*, 1992], and RF [Ho, 1995] are also used for comparison. For these 9 OB approaches, we only report the best results due to page size limitation.

Existing Attention Mechanisms

To demonstrate the effectiveness of the designed cross-attention mechanism, we have selected two state-of-the-art attention mechanisms: the Global-Attention (GA) [Zhang *et al.*, 2019] and Recurrent Attention (RA) [Qin *et al.*, 2017], as baselines.

GA and RA are the two best performing attention mechanisms from different perspectives. In our experiments, we applied GA and RA to the SFCN model and create GA-SFCN and RA-SFCN methods. Furthermore, the basic SFCN without any attention mechanism is also used as a baseline. All the three approaches, SFCN, GA-SFCN, and RA-SFCN, are compared with our proposed CA-SFCN in the experiments.

4.2 Experimental Settings

(1) Datasets: 14 real-world datasets are used to test the performance of the proposed approaches [Dua and Graff, 2017; Karim *et al.*, 2019] We utilize datasets with at least 128 instances because the batch size in the FCN model training is set to be 128, as same as the batch size in [Karim *et al.*, 2019]. For datasets with less than 128 instances, we only pick the one with the largest number of instances as an example ("HT Sensor"). Table 1 shows the detailed statistics for the datasets. **(2) Evaluation measurements:** We report the classification accuracy to show the performance. **(3) Parameter setting:** The convolutional and pooling layers use the similar configuration as that in [Karim *et al.*, 2019]. In particular, the convolutional layers contain three 2-D layers with filter sizes $8 * 1$, $5 * 1$, and $3 * 1$, the corresponding filter numbers for the three layers are 128, 256, and 128. **(4) Source code:** The source code can be found from https://github.com/huipingcao/nmsu_yhao_ijcai2020.

4.3 Effectiveness Analysis

This section shows the results of our proposed method and other approaches on different datasets. Table 2 presents the classification accuracy on classifying the datasets. Note that the settings of the convolutional layers are the same for all the methods (except the "Best-of-OB") for this set of experiments.

Effectiveness using Same Convolutional Configuration

To better compare the performance of the different approaches, the 14 datasets are split into two groups: (i) the first group contains seven datasets, on which the best accuracy that the existing methods can achieve is lower than 0.95. Existing methods do not show good performance on these datasets because they cannot capture the intrinsic features hidden in those datasets. The results on these seven dataset are reported in the first seven rows of Table 2. (ii) the second group contains the remaining seven datasets, on which the existing method can achieve the best accuracy above 0.95. The results on these seven dataset are reported in the last seven rows of Table 2. The room of improving the classification performance is higher for the first group of data than the second group of data. Table 2 shows the classification accuracy of different approaches on the two groups of datasets. The highest accuracy is highlighted.

As we can see from Table 2, the proposed approach, CA-SFCN, achieves the highest accuracy on all datasets except one (Ozone) from the first group. The result of our method on the Ozone dataset is also comparable to other baselines. Compared with the second-best method, CA-SFCN improves the averaged accuracy by around 2%. Compared with the state-of-the-art methods, CA-SFCN gets competitively similar performance on the second group datasets. Even the SFCN without any attention mechanism can classify those datasets well. This means that the instances belonging to different classes in these datasets are relatively easier to be separated. The results show that our method can achieve better improvement on datasets where previous works generate poorer results.

Our further analysis focuses on the performance on the first group of datasets. The last four columns in Table 2 specifically show the effectiveness of the designed cross-attention mechanism. Using the same parameter configurations, SFCNs with attentions (GA, RA, or CA) achieve better

Dataset	Methods					
	$SFCN_1$	$CA\text{-}SFCN_1$	$SFCN_3$	$CA\text{-}SFCN_3$	$SFCN_5$	$CA\text{-}SFCN_5$
Action	0.819	0.835	0.808	0.835	0.811	0.829
Activity	0.600	0.617	0.606	0.623	0.612	0.623
Eeg	0.547	0.578	0.547	0.656	0.594	0.609
Eeg2	0.980	0.985	0.978	0.983	0.971	0.977
Ges	0.525	0.545	0.561	0.591	0.570	0.561
HT Sensor	0.800	0.800	0.800	0.800	0.820	0.820
Ozone	0.803	0.809	0.786	0.792	0.809	0.821

* The subscript $_k$ in $SFCN_k$ and $CA\text{-}SFCN_k$ denotes the number of convolutional layers

Table 3: The effect of the proposed CA mechanism for different settings of convolutional layers in SFCN

average performance than SFCN without any attention.

Among those three different attention mechanisms, the CA-SFCN outperforms the other two attention mechanisms, GA-SFCN and RA-SFCN by around 2% to 3% respectively. The results support our design rationale that the newly designed CA can better capture the MTS features.

Effectiveness using Different Convolutional Configurations

We further explore the effectiveness of the CA mechanism with different configurations of the convolutional layers.

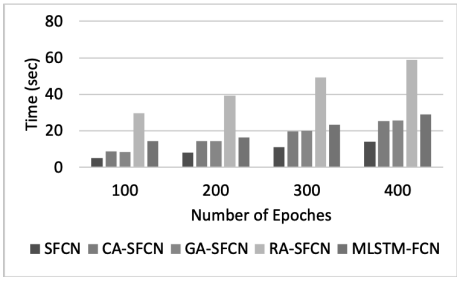
We built three SFCN models with 1, 3, and 5 convolutional layers. These SFCN models are denoted as $SFCN_1$, $SFCN_3$, and $SFCN_5$. $SFCN_1$ has one convolutional layer with filter size $8 * 1$; $SFCN_3$ has three convolutional layer with filter size $8*1$, $5*1$, and $3*1$, the same as the SFCN configuration in Table 2. $SFCN_5$ has two more convolutional layer (with filter sizes $3 * 1$ and $3 * 1$) after $SFCN_3$.

Table 3 presents the results of $SFCN_k$ and $CA\text{-}SFCN_k$ with k convolutional layers. $CA\text{-}SFCN_k$ is the $SFCN_k$ model with the cross-attention mechanism. The performances of $SFCN_k$ models can be improved by around 2% by applying the CA mechanism. The results show that the proposed CA mechanism can advance the performance of different convolutional models. Different configurations of SFCN show different accuracy values. The optimal configuration for SFCN is still an open question and that is not the focus of this paper.

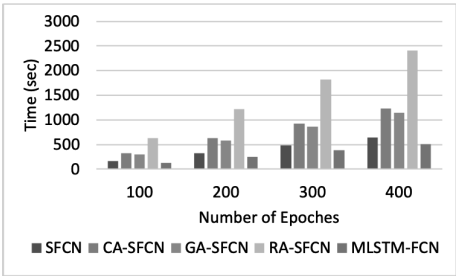
4.4 Efficiency Analysis

This section shows the training time of the proposed CA-SFCN and other baselines. We focus on the efficiency analysis of models with attention mechanisms (CA-SFCN, GA-SFCN, and RA-SFCA) because the newly designed CA mechanism is the major contribution of this work. Their performance is compared with SFCN to show the differences in model training with and without attention mechanisms. MLSTM-FCN is also used in this comparison since MLSTM-FCN achieves the second-best performance in Table 2. We show the running time on two representative datasets, including the dataset with a small number of instances (Eeg) and the dataset with a large number of instances (Har). Fig 3 shows the training time of the different models on the two datasets.

Let us first compare the training efficiency of the four SFCN-based models. The figures show that SFCN can be trained faster than the SFCN models with attention mechanisms since it does not calculate any attention. CA-SFCN and GA-SFCN show similar performance on both datasets because both CA and GA calculate the pair-wise dependencies between all feature values. RA-SFCN is the least efficient model due to the sequential calculation in RA.



(a) Eeg



(b) HAR

Figure 3: Training time (sec) on different datasets

The results also show that MLSTM-FCN runs faster than CA-SFCN on HAR. It is because MLSTM-FCN combines all the variables at the first convolutional layer (using the Conv1D operation), but SFCN model uses Conv2D operations. The running time difference is not significant on small datasets (e.g., Eeg). The results demonstrate that CA is an effective and efficient attention mechanism for identifying features from and classify MTS data.

5 Conclusions

This paper presents a new attention mechanism, cross-attention (CA), and a CA-based stabilized fully convolutional network (CA-SFCN) to classify MTS data. The newly proposed CA captures the long- and short-term dependencies of values in times series and the interactions of the multiple variables in the MTS by integrating the temporal attention (TA) and variable attention (VA). To the best of our knowledge, CA is the first attention mechanism to leverage both the temporal connection and pair-wise variable dependencies on MTS data. The experiments on 14 real-world datasets show that the proposed model outperforms the state-of-the-art models.

Acknowledgments

This work is supported by NSF #1633330, #1345232, #1914635, and #1757207.

References

[Bahdanau *et al.*, 2015] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. In *3rd ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, 2015.

[Baydogan and Runger, 2015] Mustafa Gokce Baydogan and George C. Runger. Learning a symbolic representation for multivariate time series classification. *Data Min. Knowl. Discov.*, 29(2):400–422, 2015.

[Baydogan and Runger, 2016] Mustafa Gokce Baydogan and George C. Runger. Time series representation and similarity based on local autopatterns. *Data Min. Knowl. Discov.*, 30(2):476–509, 2016.

[Boser *et al.*, 1992] Bernhard E. Boser, Isabelle M. Guyon, and Vladimir N. Vapnik. A training algorithm for optimal margin classifiers. In *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, COLT, pages 144–152. ACM, 1992.

[Dua and Graff, 2017] Dheeru Dua and Casey Graff. UCI machine learning repository, 2017.

[Hao *et al.*, 2019] Yifan Hao, Huiping Cao, Abdullah Mueen, and Sukumar Brahma. Identify significant phenomenon-specific variables for multivariate time series. *IEEE Trans. Knowl. Data Eng.*, 31(1):1–13, 2019.

[Ho, 1995] Tin Kam Ho. Random decision forests. In *Proceedings of the 3rd ICDAR Third International Conference on Document Analysis and Recognition (Volume 1) - Volume 1*, ICDAR '95, pages 278–. IEEE Computer Society, 1995.

[Ioffe and Szegedy, 2015] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, pages 448–456, 2015.

[Karim *et al.*, 2018] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Shun Chen. LSTM fully convolutional networks for time series classification. *IEEE Access*, 6:1662–1669, 2018.

[Karim *et al.*, 2019] Fazle Karim, Somshubra Majumdar, Houshang Darabi, and Samuel Harford. Multivariate lstm-fcns for time series classification. *Neural Networks*, 116:237–245, 2019.

[Long *et al.*, 2015] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*, pages 3431–3440, 2015.

[Miyato *et al.*, 2018] Takeru Miyato, Toshiki Kataoka, Masanori Koyama, and Yuichi Yoshida. Spectral normalization for generative adversarial networks. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*, 2018.

[Pascanu *et al.*, 2014] Razvan Pascanu, Çaglar Gülçehre, Kyunghyun Cho, and Yoshua Bengio. How to construct deep recurrent neural networks. In *2nd International Conference on Learning Representations*, 2014.

[Pei *et al.*, 2018] Wenjie Pei, Hamdi Dibeklioglu, David M. J. Tax, and Laurens van der Maaten. Multivariate time-series classification using the hidden-unit logistic model. *IEEE Trans. Neural Netw. Learning Syst.*, 29(4):920–931, 2018.

[Qin *et al.*, 2017] Yao Qin, Dongjin Song, Haifeng Chen, Wei Cheng, Guofei Jiang, and Garrison W. Cottrell. A dual-stage attention-based recurrent neural network for time series prediction. In *Proceedings of the 26th IJCAI 2017, Melbourne, Australia, August 19-25, 2017*, pages 2627–2633, 2017.

[Schäfer and Leser, 2017] Patrick Schäfer and Ulf Leser. Multivariate time series classification with WEASEL+MUSE. *CoRR*, abs/1711.11343, 2017.

[Seto *et al.*, 2015] Skyler Seto, Wenyu Zhang, and Yichen Zhou. Multivariate time series classification using dynamic time warping template selection for human activity recognition. In *IEEE Symposium Series on Computational Intelligence, SSCI*, pages 1399–1406, 2015.

[Tuncel and Baydogan, 2018] Kerem Sinan Tuncel and Mustafa Gokce Baydogan. Autoregressive forests for multivariate time series modeling. *Pattern Recognition*, 73:202–215, 2018.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, 4-9 December 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[Wang *et al.*, 2017] Zhiguang Wang, Weizhong Yan, and Tim Oates. Time series classification from scratch with deep neural networks: A strong baseline. In *2017 International Joint Conference on Neural Networks, IJCNN 2017, Anchorage, AK, USA, May 14-19, 2017*, pages 1578–1585, 2017.

[Yang *et al.*, 2015] Jianbo Yang, Minh Nhut Nguyen, Phyo Phyo San, Xiaoli Li, and Shonali Krishnaswamy. Deep convolutional neural networks on multichannel time series for human activity recognition. In *Intl. Joint Conference on Artificial Intelligence, IJCAI*, pages 3995–4001, 2015.

[Yoon *et al.*, 2005] Hyunjin Yoon, Kiyoung Yang, and Cyrus Shahabi. Feature subset selection and feature ranking for multivariate time series. *IEEE Trans. Knowl. Data Eng.*, 17(9):1186–1198, 2005.

[Zhang *et al.*, 2019] Han Zhang, Ian J. Goodfellow, Dimitris N. Metaxas, and Augustus Odena. Self-attention generative adversarial networks. In *Proceedings of the 36th International Conference on Machine Learning, ICML 2019, 9-15 June 2019, Long Beach, California, USA*, pages 7354–7363, 2019.