MRM2: Multi-Relationship Modeling Module for Multivariate Time Series Classification

Pengxiang Shi*, Xuan Dang*, Wenwen Ye†, Zhipeng Li‡, Zheng Qin§
*§*School of Software,Tsinghua University*,Beijing, China,{shipx19,dx18,qingzh}@mails.tsinghua.edu.cn
†*School of Software,Tsinghua University*,Beijing, China,zhipenglithu@163.com
‡*Baidu*,Beijing, China,two_ye@hotmail.com

***Abstract*—Multivariate Time Series Classification (MTSC) is a prevalent but challenging problem in data mining. With the development of Deep Neural Networks (DNN), hundreds of deep models for MTSC have been proposed. However, most prior works only explicitly model the relationship between time series and classes and ignore the diversity of the relationship, suffering from insufficient information exploitation. In this paper, we propose a novel module named Multi-Relationship Modeling Module(MRM2) for more effective MTSC. MRM2 uses the classified labels to explicitly model not only the relationship between time series and classes, but also the relationship among time series, enabling the backbone to generate distinguishable embeddings. In addition, MRM2 is versatile because it can be combined with the existing backbones of DNN for end-to-end training. Finally, we conduct a series of ablation studies and comparative experiments on the real multivariate time series archive UEA. Experimental results indicate that MRM2 can significantly improve classification performance in most cases. Codes are available on GitHub [1].**

***Index Terms*—time series, classification, deep neural networks, multi-relationship**

I. Introduction

With the development of deep learning technology, more and more works use DNN to solve the time series classification. Various deep models have been developed for time series, such as 1D-Residual Network(ResNet) [15], Long Short-Term Memory(LSTM) [9], and self-attention models [10]. These works are single-relationship oriented, which are developed to only explicitly model the Relationship between Time series and Classes(TCR). However, in addition to TCR, modeling the Relationship among Time series(TTR) can also solve the MTSC task. For instance, traditional distance-based models [12, 11] mainly focus on modeling the TTR, rather than the TCR, but also achieve robust and advanced results on MTSC, which shows that TTR, like the TCR, is beneficial information for MTSC. This motivates us to investigate how to leverage both the TCR and TTR relationships during training to enhance the multivariate time series classification. In order to fully explore and leverage the information of multi-relationship for MTSC models, there are still two main challenges to solve:

Firstly, the modeling target of TCR is not naturally consistent with the modeling target of TTR. TCR modeling aims to predict the class to which the time series belongs, while TTR modeling aims to predict the similarity between two time series. Due to the different modeling targets, it will be difficult to make TCR and TTR modeling benefit each other and improve the overall performance. Therefore, how to utilize the same supervision information to model both TCR and TTR while avoiding inconsistencies in modeling targets is still a great challenge. Secondly, models that model both TCR and TTR should not significantly increase the number of parameters. Due to the limited size of labeled time series dataset, models with many parameters are prone to overfitting, resulting in performance degradation. Therefore, how to simultaneously model TCR and TTR with limited parameters is another challenge.

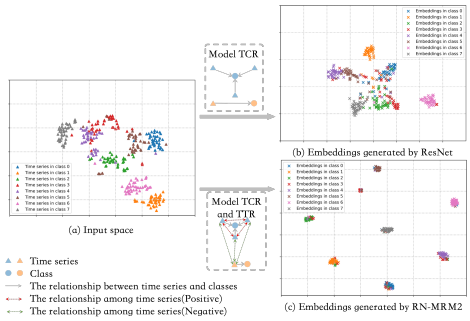


Fig. 1. The advantage of modeling both TCR and TTR via MRM2. Only explicitly modeling TCR cannot yield distinct distinguishable embeddings, but explicitly modeling both TCR and TTR enables the backbone to generate clear distinguishable embeddings.

To handle the above issues, in this paper, we propose a novel multivariate time series classification module named **M**ulti-**R**elationship **M**odeling **M**odule(MRM2), which contains two components: 1) **T**ime series and **C**lasses **R**elationship **M**odeling **M**odule(TCRM2); 2) **T**ime series and **T**ime series **R**elationship **M**odeling **M**odule(TTRM2). Fig. 1 shows the advantage of modeling both TCR and TTR via MRM2. The time series used in Fig. 1 is the test set of UWaveGestureLibrary[2], and the backbone network is ResNet [15]. Fig. 1(a) is a visualization of the raw time series using t-SNE, Fig. 1(b) is the embeddings visualization when only explicitly modeling TCR, and Fig. 1(c) is the embeddings

* These authors contributed equally.
§ is corresponding authors
[1]https://github.com/spx1997/MRM2

[2]This dataset from multivariate time series archive UEA

visualization when explicitly modeling both TCR and TTR. From Fig. 1, we can find that modeling both TCR and TTR can generate discriminative embeddings.In order to further improve the classification performance, we extend TTRM2 to semi-supervised settings and design **Semi**-supervised **TTRM2** (Semi-TTRM2). Semi-TTRM2 will model not only the relationship among labeled time series, but also the relationship between labeled and unlabeled time series.

Our main contributions are summarized as follows:

(1) In this paper, we present a focused study on multi-relationship modeling for multivariate time series classification. To the best of our knowledge, this is among the first few studies to investigate how to model multi-relationship for better MTSC performance.

(2) We design a novel module MRM2, consisting of two components: TCRM2 and TTRM2. In order to use unlabeled data, we extend TTRM2 to semi-supervised settings and design the Semi-TTRM2. Semi-TTRM2 can model the relationship between labeled time series and unlabeled time series.

(4) We conduct extensive ablation studies and comparative experiments in multivariate time series archive UEA to evaluate MRM2 from various angles. The results demonstrate that our proposed MRM2 can significantly improve the classification performance in most cases.

II. Related Works

A. The Traditional Models

Multivariate time series classification is a fundamental task in multivariate time series mining, and there are many related works. Based on the univariate time series classification model DTW-KNN, two models are designed to deal with MTSC: DTW$_I$-KNN [12] and DTW$_D$-KNN [12]. Although DTW$_{I/D}$-KNN is time-consuming, it is still a very competitive benchmark due to its simplicity, robustness, and not requiring extensive hyperparameter settings.

B. The Deep Learning-based Models

Inspired by the deep learning model in computer vision, some works [2, 8, 17] change the two dimensional convolution to the one dimensional for series processing, and this model showed exciting results on TSC. The work [15] introduces the Full Convolution Network(FCN) and residual connection into the TSC or MTSC. Some works [3, 13] optimize the classification results by combining different convolution kernels. The work [1] introduces the dilated convolution and causal convolution into the convolution neural network to alleviate the problem that the convolution can not capture long distance dependence. In order to further expand the ability of models that capture long distance context dependence, some researchers use Recurrent Neural Network(RNN) [5] and LSTM [9] to solve the task of time series classification. In addition, some works [6, 7] are inspired by ensemble learning, and combine RNN and CNN to further improve the model's ability that extracts time series features.

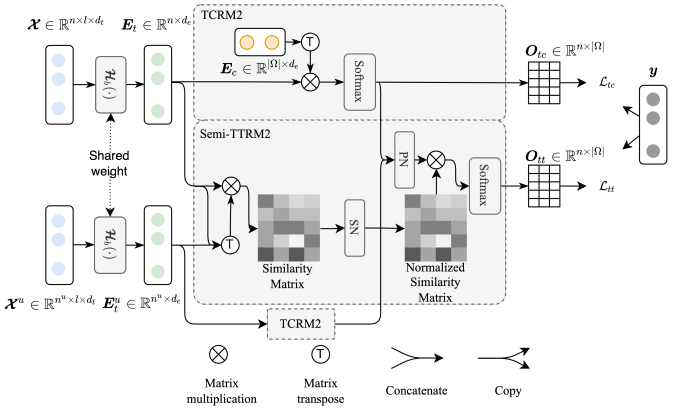


Fig. 2. Overall framework of the MRM2. MRM2 includes two components: TCRM2 and Semi-TTRM2.

The deep models introduced above only explicitly model TCR. In the field of time series classification, there are few works focus on simultaneously modeling TCR and TTR. Though there are some similar works in other fields, they are essentially different from our method. To be specific, Center Loss [16] in computer vision has a similar purpose to our proposed Semi-TTRM2, *i.e.*, to generate discriminative embeddings. However, Semi-TTRM2 and Center Loss are implemented in completely different ways. Semi-TTRM2 can simultaneously model the relationship among all labeled time series, as well as the relationship between labeled time series and unlabeled time series, but Center Loss is still modeling the relationship between time series and classes essentially.

III. Preliminaries

In this section, we formally introduce the symbols and terminologies. A multivariate time series is an ordered set by time, the elements in the set consists of the time and corresponding value. It can be recorded as $\boldsymbol{X} = \{\langle t_1, \boldsymbol{x}_1\rangle, ..., \langle t_l, \boldsymbol{x}_l\rangle\}$, where l is the length of time series and $\boldsymbol{x}_i \in \mathbb{R}^{d_t}$, d_t is the dimension of time series. When the intervals in the time series are the same $\Delta t = t_{i+1} - t_i$, it is usually simplified to: $\boldsymbol{X} = [\boldsymbol{x}_1, ..., \boldsymbol{x}_l] \in \mathbb{R}^{l \times d_t}$. We use $\boldsymbol{X}[i]$ to represent the value of $\boldsymbol{X}$ at t_i, *i.e.*, $\boldsymbol{X}[i] = \boldsymbol{x}_i \in \mathbb{R}^{d_t}$. A dataset composed of n time series is recorded as $\boldsymbol{\mathcal{X}} = \{\boldsymbol{X}_1, ..., \boldsymbol{X}_n\} \in \mathbb{R}^{n \times l \times d_t}$. The i-th time series in dataset $\boldsymbol{\mathcal{X}}$ can be denoted as $\boldsymbol{\mathcal{X}}[i] \in \mathbb{R}^{l \times d_t}$ or $\boldsymbol{X}_i \in \mathbb{R}^{l \times d_t}$. If the dataset has the corresponding label vector $\boldsymbol{y} = \{y_1, \cdots, y_n\} \in \mathbb{R}^n$, we mark them as $\langle \boldsymbol{\mathcal{X}}, \boldsymbol{y}\rangle$, where $\boldsymbol{y}[i] = y_i \in \Omega$ is the label of the time series $\boldsymbol{X}_i$, from a predefined label set Ω.

Our **research problem** is to accurately predict a class label of a time series by a deep neural network $\boldsymbol{\mathcal{H}}(\cdot)$. We denote the backbone of $\boldsymbol{\mathcal{H}}(\cdot)$ as $\boldsymbol{\mathcal{H}}_b(\cdot)$, and the output of $\boldsymbol{\mathcal{H}}_b(\cdot)$ are the embeddings $\boldsymbol{E}_t \in \mathbb{R}^{n \times d_e}$. The embedding of $\boldsymbol{\mathcal{X}}[i]$ is denoted as $\boldsymbol{E}_t[i] \in \mathbb{R}^{d_e}$. We will use superscript ^{u} to represent the unlabeled ones, *e.g.*, $\boldsymbol{\mathcal{X}}^u$ represents the unlabeled time series dataset. When the input is $\boldsymbol{\mathcal{X}}^u$, $\boldsymbol{E}_t^u$ is used to represent the output of $\boldsymbol{\mathcal{H}}_b(\cdot)$.

IV. Approach

In this section, we will introduce the framework of our proposed MRM2. As shown in Fig. 2, MRM2 includes two components: TCRM2 and TTRM2. We will explain them separately in the following part.

A. TCRM2

Before modeling the TCR, it is necessary to answer what relationship exists between time series and classes. When both time series and classes are in the embedding space, a given time series' embedding should be similar to the embedding of the class to which it belongs, and far away from the embeddings of other classes. Therefore, we can model TCR from this perspective. The embeddings of time series can be obtained through the backbone network $\boldsymbol{\mathcal{H}}_b(\cdot)$, so the key to TCR modeling is how to obtain proper class embeddings.

As shown in Fig. 2, we denote the time series embeddings generated by backbone network as $\boldsymbol{E}_t \in \mathbb{R}^{n\times d_e}$, the class embeddings as $\boldsymbol{E}_c \in \mathbb{R}^{|\Omega|\times d_c}$. $\boldsymbol{E}_c$ are trainable parameters. The dimension of time series embeddings is equal to the dimension of class embeddings, *i.e.*, $d_e = d_c$. $\boldsymbol{S}_{tc} \in \mathbb{R}^{n\times|\Omega|}$ is marked as the matrix multiplication of $\boldsymbol{E}_t$ and $\boldsymbol{E}_c$, which is used to express the similarity between time series and classes, namely $\boldsymbol{S}_{tc} = \boldsymbol{E}_t\boldsymbol{E}_c^T$. After softmax function, we get the output $\boldsymbol{O}_{tc} \in \mathbb{R}^{n\times|\Omega|}$ of the TCRM2:

$$\boldsymbol{O}_{tc} = \mathrm{Softmax}(\boldsymbol{S}_{tc}, 1) \ . \tag{1}$$

where $\mathrm{Softmax}(\cdot, 1)$ means to perform the softmax function on dimension 1 of input matrix.

Given a time series $\boldsymbol{\mathcal{X}}[i]$, $\boldsymbol{O}_{tc}[i]$ can be regarded as the probability vector of $\boldsymbol{\mathcal{X}}[i]$ classified into each class in Ω. Assuming that the class of $\boldsymbol{\mathcal{X}}[i]$ is $\boldsymbol{y}[i]$, then we want to maximize the similarity between time series embeddings $\boldsymbol{E}_t[i] \in \mathbb{R}^{1\times d_e}$ and class embeddings $\boldsymbol{E}_c[\boldsymbol{y}[i]] \in \mathbb{R}^{1\times d_c}$, *i.e.*, we want to maximize the probability probability $\boldsymbol{O}_{tc}[i, \boldsymbol{y}[i]]$. We use the logarithmic loss function as the loss function of TCRM2 and get:

$$\boldsymbol{\mathcal{L}}_{\mathrm{TCRM2}} = -\frac{1}{n}\sum_i^n \log(\boldsymbol{O}_{tc}[i, \boldsymbol{y}[i]]) \ . \tag{2}$$

B. TTRM2

Models such as $\mathrm{DTW}_{I/D}$-KNN[12] and EE_{ID}[11] classify test time series by modeling TTR, rather than modeling TCR. These models also achieve advanced and robust results. This shows that TTR, like TCR, is also very important information for the time series classification. This motivates us to design a module(TTRM2) to model TTR. The process of TTRM2 is shown in Fig. 3. Given a time series dataset $\boldsymbol{\mathcal{X}}$, after the backbone network and TCRM2, we will get the time series embedding $\boldsymbol{E}_t \in \mathbb{R}^{n\times d_e}$ and our prediction results $\boldsymbol{O}_{tc} \in \mathbb{R}^{n\times|\Omega|}$. We use the matrix multiplication of the time series embedding $\boldsymbol{E}_t$ and itself to represent the similarity between time series and time series $\boldsymbol{S}_{tt} = \boldsymbol{E}_t\boldsymbol{E}_t^T$, where $\boldsymbol{S}_{tt} \in \mathbb{R}^{n\times n}$ is the similarity matrix. Then we need to calculate the time series weight by similarity matrix $\boldsymbol{S}_{tt}$. If the similarity matrix $\boldsymbol{S}_{tt}$ is directly used as the weight, which will cause problems such as excessive weight. In order to solve this problem, we design a Similarity Normalization (SN) to adjust the similarity matrix $\boldsymbol{S}_{tt}$, and then use the normalized similarity matrix $\overline{\boldsymbol{S}}_{tt} \in \mathbb{R}^{n\times n}$ as the weight. In SN, the calculation formula of $\overline{\boldsymbol{S}}_{tt}$ is as follows:

$$\overline{\boldsymbol{S}}_{tt} = \frac{\boldsymbol{S}_{tt}}{\boldsymbol{\sigma}+\epsilon} \ . \tag{3}$$

where $\boldsymbol{\sigma} \in \mathbb{R}^{n\times 1}$ is the standard deviation of $\boldsymbol{S}_{tt}$, namely $\boldsymbol{\sigma}[i] = \sqrt{\frac{1}{n}\sum_j(\boldsymbol{S}_{tt}[i,j] - \frac{1}{n}\sum_j \boldsymbol{S}_{tt}[i,j])^2}$.

After obtaining the time series weight matrix $\overline{\boldsymbol{S}}_{tt}$, TTRM2 will weighted sum the prediction results $\boldsymbol{O}_{tc}$ in TCRM2 according to $\overline{\boldsymbol{S}}_{tt}$:

$$\boldsymbol{O}_{tt} = \mathrm{Softmax}(\overline{\boldsymbol{S}}_{tt}\frac{\boldsymbol{O}_{tc}}{\sum_{i=0}^n \boldsymbol{O}_{tc}[i]}, 1) \ . \tag{4}$$

where $\boldsymbol{O}_{tt} \in \mathbb{R}^{n\times|\Omega|}$ is the output of TTRM2. It should be noted that $\frac{1}{\sum_{i=0}^n \boldsymbol{O}_{tc}[i]} \in \mathbb{R}^{1\times|\Omega|}$ is indispensable, which can prevent the harmful effects caused by imbalanced classes. We name this operation Prediction Normalization (PN). In essence, TTRM2 uses the similarity between time series $\boldsymbol{\mathcal{X}}[i]$ and other time series $\boldsymbol{\mathcal{X}}[j]$ as weights, and performs weighted average on the prediction classes of other time series to obtain the result of $\boldsymbol{\mathcal{X}}[i]$, where $i, j \in \{1, \cdots, n\}$. Like the output $\boldsymbol{O}_{tc}$ of TCRM2, $\boldsymbol{O}_{tt}[i]$ also can be regarded as the probability vector of time series $\boldsymbol{\mathcal{X}}[i]$ classified into each class in Ω. Assuming that the class of $\boldsymbol{\mathcal{X}}[i]$ is $\boldsymbol{y}[i]$, we also want to maximize the probability $\boldsymbol{O}_{tt}[i, \boldsymbol{y}[i]]$. When using the logarithmic loss function, the loss function of TTRM2 is as follows:

$$\boldsymbol{\mathcal{L}}_{\mathrm{TTRM2}} = -\frac{1}{n}\sum_i^n \log(\boldsymbol{O}_{tt}[i, \boldsymbol{y}[i]]) \ . \tag{5}$$

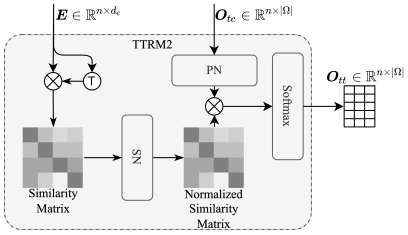


Fig. 3. Overall framework of the TTRM2.

C. Semi-TTRM2

TTRM2 introduced in the section IV-B only models the relationship among labeled time series. In this section, based on TTRM2, we design a Semi-TTRM2, which can model not only the relationship among labeled time series, but also the relationship between labeled and unlabeled time series.

As shown in Fig. 2, the unlabeled time series dataset $\boldsymbol{\mathcal{X}}^u \in \mathbb{R}^{n^u\times l\times d_t}$ passes through the backbone network $\boldsymbol{\mathcal{H}}_b(\cdot)$ to obtain the corresponding embedding $\boldsymbol{E}_t^u \in \mathbb{R}^{n^u\times d_e}$. Then through TCRM2, we get the predicted probability $\boldsymbol{O}_{tc}^u \in \mathbb{R}^{n^u\times|\Omega|}$ of the unlabeled time series $\boldsymbol{\mathcal{X}}^u$ classified into

each class. In order to simultaneously model the relationship between labeled and unlabeled time series in the training phase, we consider both labeled and unlabeled time series to calculate the similarity matrix $\boldsymbol{S}_{tt}^{semi} \in \mathbb{R}^{n\times(n+n^u)}$. The calculation formula is as follows:

$$\boldsymbol{S}_{tt}^{semi} = \boldsymbol{E}_t \begin{bmatrix} \boldsymbol{E}_t \\ \boldsymbol{E}_t^u \end{bmatrix}^T \ . \tag{6}$$

$\begin{bmatrix} \boldsymbol{E}_t \\ \boldsymbol{E}_t^u \end{bmatrix} \in \mathbb{R}^{(n+n^u)\times d_e}$ means concatenate on dimension zero. We still use SN to obtain the normalized similarity matrix $\overline{\boldsymbol{S}}_{tt}^{semi} \in \mathbb{R}^{n\times(n+n^u)}$ like TTRM2.

Finally, for each time series, we weighted sum both labeled time series predication results $\boldsymbol{O}_{tc}$ and unlabeled time series prediction results $\boldsymbol{O}_{tc}^u$ to obtain the prediction output $\boldsymbol{O}_{tt}^{semi} \in \mathbb{R}^{n\times|\Omega|}$ of the Semi-TTRM2:

$$\boldsymbol{O}_{tt}^{semi} = \mathrm{Softmax}(\overline{\boldsymbol{S}}_{tt}^{semi} \frac{\begin{bmatrix} \boldsymbol{O}_{tc} \\ \boldsymbol{O}_{tc}^u \end{bmatrix}}{\sum_{i=0}^{n+n^u} \begin{bmatrix} \boldsymbol{O}_{tc} \\ \boldsymbol{O}_{tc}^u \end{bmatrix}^{[i]}}, 1) \ . \tag{7}$$

Essentially, Semi-TTRM2 uses the similarity between time series $\boldsymbol{\mathcal{X}}[i]$ and other time series (including labeled time series and unlabeled time series) $\boldsymbol{\mathcal{X}}[j]$ as weights, then gets the result of time series $\boldsymbol{\mathcal{X}}[i]$ by the weighted average other time series prediction results, where $i \in \{1, \cdots, n\}$ and $j \in \{1, \cdots, n+n^u\}$. Like the output $\boldsymbol{O}_{tt}$ of TTRM2, $\boldsymbol{O}_{tt}^{semi}[i]$ also can be regarded as the probability vector of time series $\boldsymbol{\mathcal{X}}[i]$ classified into each class in Ω. Assuming that the class of $\boldsymbol{\mathcal{X}}[i]$ is $\boldsymbol{y}[i]$, we also want to maximize the probability $\boldsymbol{O}_{tt}^{semi}[i, \boldsymbol{y}[i]]$, so we still using the logarithmic loss function as the loss function of Semi-TTRM2 to provide more supervision information.

$$\boldsymbol{\mathcal{L}}_{\mathrm{Semi-TTRM2}} = -\sum_i^n \log(\boldsymbol{O}_{tt}^{semi}[i, \boldsymbol{y}[i]]) \ . \tag{8}$$

D. Total Loss Function

In the previous section, we introduced TCRM2, TTRM2 and Semi-TTRM2, respectively. The functions of Semi-TTRM2 and TTRM2 are similar, and TTRM2 is a special case of Semi-TTRM2. In subsequent experiments, our proposed MRM2 include TCRM2 and Semi-TTRM2, unless stated otherwise. The overall framework of MRM2 is shown in Fig. 2. The total loss function in the training phase is as follows:

$$\boldsymbol{\mathcal{L}} = \boldsymbol{\mathcal{L}}_{\mathrm{TCRM2}} + \lambda \boldsymbol{\mathcal{L}}_{\mathrm{Semi-TTRM2}} \ . \tag{9}$$

where λ is hyperparameter, we will determine them through experiments.

V. Experimental Result

In this section, we conduct experiments to verify the performance of MRM2, aiming to answer the following research questions:

- **RQ1**: How much does MRM2 help to improve the classification performance of the deep model?
- **RQ2**: Compared with other models that can also model both TCR and TTR, does MRM2 have any advantages?

A. Settings

1) Implementation Details: The models are performed on one Nvidia Tesla P40 with CUDA 11.0. The source code can be accessed on Github[3].

2) Datasets: We utilize real data to validate our model. The real data comes from UEA archive. Some datasets in UEA have too few test time series. For example, StandWalkJump and AtrialFibrillation only have 15 test samples. Since it is unreasonable to conduct experiments on datasets with too few test time series, we only carry out experiments on 22 datasets with more than 100 test time series in UEA. The more detailed information about UEA can be found on the link[4].

3) Model Settings: We implement our MRM2 on five state-of-the-art deep models, which are:**ResNet**[15][5], **FCN**[15], **MLSTM-FCN**[7], **InceptionTime** [3], and **OS-CNN**[13]. We use DM(Deep Models) to indicate these state-of-the-art deep models. We replace the last linear layer and softmax of these five deep models with MRM2, and get five new models: **RN-MRM2**, **FCN-MRM2**, **MF-MRM2**, **IT-MRM2** and **OS-MRM2**. We use DM-MRM2 to indicate these models.

4) Metrics: We report the classification accuracy Acc and the average rank $AvgRank$ to show the classification performance. In order to intuitively show the comparison between DM-MRM2 and DM, we also compute the number of $Win/Tie/Loss$ and the average absolute improvement $AvgImp_{MRM2} = \frac{1}{M}\sum_i^M (Acc_{DM-MRM2}^i - Acc_{DM}^i)$, where M is the number of datasets.

5) Training and Test: All deep models in this section are implemented by PyTorch. The code of ResNet, FCN, InceptionTime and MLSTM-FCN come from the open source repository tsai[6]. The code for OS-CNN comes from its author[7]. DM and DM-MRM2 are trained by Adam. The maximum epoch is 600, the learning rate is 0.001, and the regularization coefficient is 0.01. The function of *ReduceLROnPlateau* in PyTorch with parameter *model='min'*, *factor=0.5*, *patience=50* and *min_lr=5e*$^{-5}$ is used to adjust the learning rate. In order to make the model converge faster, DM-MRM2 only uses the TCRM2 in the first 400 training epochs. In the 400th epoch, DM-MRM2 adjusts the learning rate to 0.001, and TCRM2 and Semi-TTRM2 are used simultaneously in the last 200 epochs. To compare fairly, the learning rate of DM is also adjusted to 0.001 at the 400th epoch. The hyperparameter λ is set to 1. Following the works[15, 13, 3], we train/test the models using the original training/testing splits provided in the UEA, and save the model parameters with the lowest training loss and report their performance on the test set. In the testing phase, MRM2 only uses TCRM2 to predict the results for testing efficiency.

[3]https://github.com/spx1997/MRM2

[4]http://www.timeseriesclassification.com/Downloads/Archives/Multivariate2018_arff.zip

[5]This network structure is differ from ResNet[4] in field of computer vision.

[6]https://github.com/timeseriesAI/tsai

[7]https://github.com/Wensi-Tang/OS-CNN

TABLE I
COMPARISON OF DM AND DM-MRM2 ON UEA. BOLD INDICATES THE BEST PERFORMING METHODS.

Dataset	FCN	FCN-MRM2	MLSTM-FCN	MF-MRM2	ResNet	RN-MRM2	Inception Time	IT-MRM2	OS-CNN	OS-MRM2
ArticularyWordRecognition	0.985	**0.988**	0.989	**0.993**	**0.988**	**0.988**	0.987	**0.988**	0.989	**0.991**
CharacterTrajectories	**0.988**	**0.988**	**0.990**	**0.990**	0.986	**0.988**	**0.995**	**0.995**	**0.995**	**0.995**
EigenWorms	0.589	**0.887**	0.195	**0.882**	0.519	**0.693**	0.634	**0.768**	0.431	**0.908**
Epilepsy	0.977	**0.987**	0.980	**0.988**	**0.978**	**0.978**	**0.977**	**0.977**	**0.984**	**0.984**
EthanolConcentration	0.299	**0.312**	0.339	**0.368**	0.287	**0.301**	0.273	**0.310**	0.283	**0.317**
ERing	0.890	**0.963**	0.899	**0.955**	0.836	**0.857**	0.867	**0.924**	0.901	**0.955**
FaceDetection	**0.535**	0.520	0.517	**0.528**	0.549	**0.558**	0.649	**0.660**	0.531	**0.548**
Handwriting	0.496	**0.520**	0.520	**0.524**	0.483	**0.551**	0.623	**0.635**	0.617	**0.649**
Heartbeat	0.752	**0.782**	0.727	**0.759**	0.753	**0.766**	0.759	**0.767**	0.750	**0.755**
InsectWingbeat	0.655	**0.665**	0.651	**0.659**	0.648	**0.658**	0.683	**0.691**	0.652	**0.665**
JapaneseVowels	0.989	**0.991**	0.989	**0.990**	0.983	**0.989**	0.964	**0.984**	0.988	**0.989**
Libras	0.952	**0.961**	0.948	**0.960**	0.940	**0.952**	0.876	**0.898**	0.952	**0.958**
LSST	0.608	**0.613**	**0.662**	0.639	0.697	**0.703**	0.649	**0.652**	0.611	**0.642**
NATOPS	0.952	**0.976**	0.966	**0.974**	0.958	**0.989**	0.976	**0.980**	0.960	**0.972**
PenDigits	0.985	**0.993**	0.980	**0.991**	0.985	**0.991**	0.987	**0.995**	0.977	**0.985**
PEMS-SF	0.623	**0.748**	0.727	**0.751**	0.754	**0.764**	0.736	**0.793**	**0.740**	**0.740**
PhonemeSpectra	0.244	**0.260**	**0.270**	0.260	**0.320**	0.314	0.326	**0.334**	0.319	**0.322**
RacketSports	0.874	**0.876**	0.876	**0.889**	0.851	**0.901**	0.880	**0.895**	0.871	**0.883**
SelfRegulationSCP1	**0.796**	0.771	0.755	**0.758**	**0.822**	0.800	**0.844**	0.841	0.769	**0.800**
SelfRegulationSCP2	0.513	**0.537**	0.497	**0.514**	**0.510**	0.501	0.488	**0.494**	**0.526**	0.499
SpokenArabicDigits	0.991	**0.992**	0.988	**0.994**	0.987	**0.993**	0.994	**0.999**	0.994	**0.999**
UWaveGestureLibrary	0.868	**0.874**	0.865	**0.891**	0.833	**0.867**	0.876	**0.901**	0.919	**0.941**
$AvgImp$	2.92%		4.21%		1.98%		1.99%		3.35%	
$Win/Tie/Loss$	19/1/2		19/1/2		17/2/3		19/2/1		18/3/1	

B. Comparison of DM and DM-MRM2(RQ1)

This subsection will verify that the MRM2 can help the existing deep models improve their accuracy. We report the results of the DM and DM-MRM2 in Table I. The accuracy results of DM and DM-MRM2 are the mean values of five runs. From this table, we can find that MRM2 can bring 2.92%, 4.21%, 1.98%, 1.99% and 3.35% average absolute improvement to FCN, MLSTM-FCN, ResNet, InceptionTime and OS-CNN, respectively, and at 19/22, 19/22, 17/22, 19/22 and 18/22 datasets beat the original deep models. MRM2 can stably improve the classification accuracy of various existing deep models on most time series datasets, which shows the robustness and effectiveness of MRM2.

C. Comparison of MRM2 and Center Loss/GAT(RQ2)

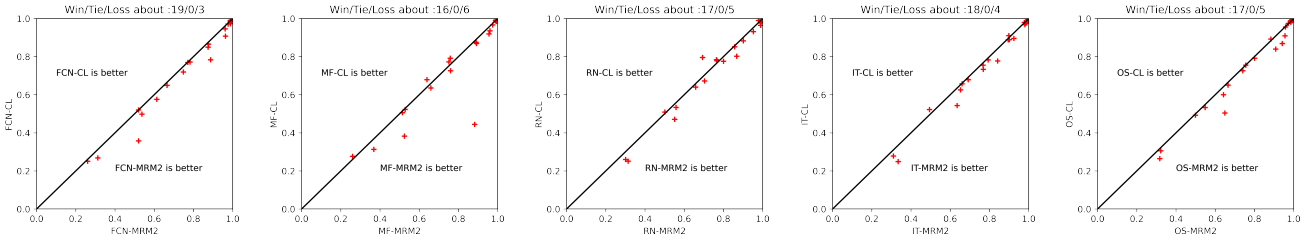
We demonstrate through extensive experiments in section V-B that MRM2 can significantly improve the classification performance of existing deep models. The reason why MRM2 works is that two relations TCR and TTR are explicitly modeled at the same time during training. In this subsection, we will show that MRM2 is superior in modeling both TCR and TTR compared to other models. We implement GAT[14] and Center Loss[16] on the backbone of the five deep models introduced in section V-A3, which we denote as DM-GAT and DM-CL. DM-GAT treat the time series within a batch as nodes on a graph, and use the similarity of the nodes (that is, the similarity of the corresponding time series embeddings) to calculate the weight of the edges in the graph. Next, we use this graph to aggregate and fuse the embedding of the time series to model the TTR. In detail, we use one graph attention layer in GAT and use residual connections to fuse the embeddings generated by the backbone and embeddings generated by GAT. The fused embeddings are used to predict results. The loss of DM-CL includes: the original classification loss and the center loss. The center loss has similar effects to TTRM2/Semi-TTRM2, making the embeddings of time series in the same class closer and the embeddings of time series in different classes farther. The implementation of center loss comes from Github[8].

We report the comparison results on Fig. 4 and Fig. 5. We find that the $Win/Tie/Loss$ of DM-MRM2 compared to DM-CL are: 19/0/3, 16/0/6, 17/0/5, 18/0/4 and 17/0/5. The $Win/Tie/Loss$ of DM-MRM2 compared to DM-GAT are: 21/0/1, 22/0/0, 19/0/3, 19/0/3 and 21/0/1. This demonstrates that the proposed DM-MRM2 is indeed superior to other models (DM-GAT and DM-CL) in explicitly modeling both TCR and TTR, and therefore has better classification performance.

VI. CONCLUSION

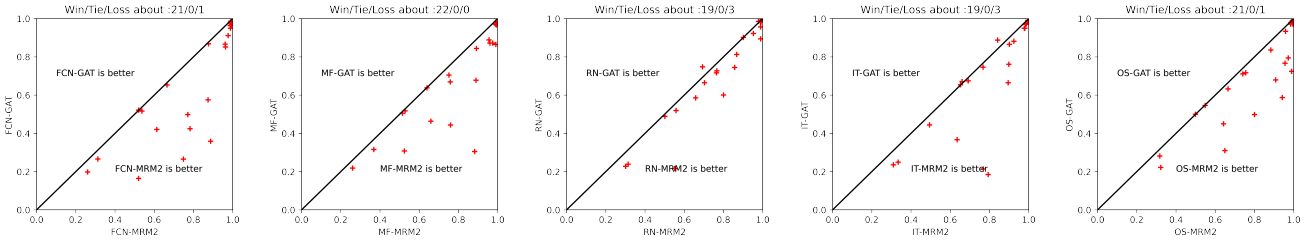
In this paper, we uncover the importance of exploring multi-relationship for multivariate time series classification and propose a novel module named MRM2, which can leverage the multi-relationship information to improve the MTSC performance. Specifically, MRM2 contains two modules, namely TCRM2 and TTRM2, which are used to model TCR and TTR, respectively. To overcome the inconsistencies in the targets of TCR and TTR, MRM2 uses the same label information to supervise TCRM2 and TTRM2, which allows them to promote each other, and enables the backbone to generate more distinguishable embeddings, thereby improves the classification performance. In order to further explore TTR, we design Semi-TTRM2 based on TTRM2. MRM2 with Semi-TTRM2 can utilize both labeled and unlabeled data to further improve the MTSC performance. Experiments on 22 real datasets have demonstrated the effectiveness of MRM2 on a variety of backbones.

[8]https://github.com/KaiyangZhou/pytorch-center-loss



(a) FCN-MRM2 VS FCN-CL (b) MF-MRM2 VS MF-CL (c) RN-MRM2 VS RN-CL (d) IT-MRM2 VS IT-CL (e) OS-MRM2 VS OS-CL

Fig. 4. Comparison of DM-MRM2 and DM-CL.



(a) FCN-MRM2 VS FCN-GAT (b) MF-MRM2 VS MF-GAT (c) RN-MRM2 VS RN-GAT (d) IT-MRM2 VS IT-GAT (e) OS-MRM2 VS OS-GAT

Fig. 5. Comparison of DM-MRM2 and DM-GAT.

REFERENCES

[1] Shaojie Bai, J Zico Kolter, and Vladlen Koltun. "An empirical evaluation of generic convolutional and recurrent networks for sequence modeling". In: *arXiv preprint arXiv:1803.01271* (2018).

[2] Zhicheng Cui, Wenlin Chen, and Yixin Chen. "Multi-scale convolutional neural networks for time series classification". In: *arXiv preprint arXiv:1603.06995* (2016).

[3] Hassan Ismail Fawaz et al. "Inceptiontime: Finding alexnet for time series classification". In: *Data Mining and Knowledge Discovery* 34.6 (2020), pp. 1936–1962.

[4] Kaiming He et al. "Deep residual learning for image recognition". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[5] Michael Hüsken and Peter Stagge. "Recurrent neural networks for time series classification". In: *Neurocomputing* 50 (2003), pp. 223–235.

[6] Fazle Karim et al. "LSTM fully convolutional networks for time series classification". In: *IEEE access* 6 (2017), pp. 1662–1669.

[7] Fazle Karim et al. "Multivariate LSTM-FCNs for time series classification". In: *Neural Networks* 116 (2019), pp. 237–245.

[8] Arthur Le Guennec, Simon Malinowski, and Romain Tavenard. "Data augmentation for time series classification using convolutional neural networks". In: 2016.

[9] Khaled Saleh, Mohammed Hossny, and Saeid Nahavandi. "Driving behavior classification based on sensor data fusion using LSTM recurrent neural networks". In: *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE. 2017, pp. 1–6.

[10] Pengxiang Shi, Wenwen Ye, and Zheng Qin. "Self-Supervised Pre-training for Time Series Classification". In: *2021 International Joint Conference on Neural Networks (IJCNN)*. IEEE. 2021, pp. 1–8.

[11] Ahmed Shifaz et al. "Elastic similarity measures for multivariate time series classification". In: *arXiv preprint arXiv:2102.10231* (2021).

[12] Mohammad Shokoohi-Yekta, Jun Wang, and Eamonn Keogh. "On the non-trivial generalization of dynamic time warping to the multi-dimensional case". In: *Proceedings of the 2015 SIAM international conference on data mining*. SIAM. 2015, pp. 289–297.

[13] W. Tang et al. "Rethinking 1D-CNN for Time Series Classification: A Stronger Baseline". In: *ArXiv* abs/2002.10061 (2020).

[14] Petar Velickovic et al. "Graph attention networks". In: *stat* 1050 (2017), p. 20.

[15] 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)*. IEEE. 2017, pp. 1578–1585.

[16] Yandong Wen et al. "A discriminative feature learning approach for deep face recognition". In: *European conference on computer vision*. Springer. 2016, pp. 499–515.

[17] Bendong Zhao et al. "Convolutional neural networks for time series classification". In: *Journal of Systems Engineering and Electronics* 28.1 (2017), pp. 162–169.