DTCM: Deep Transformer Capsule Mutual Distillation for Multivariate Time Series Classification

Zhiwen Xiao, *Member, IEEE*, Xin Xu, Huanlai Xing, *Member, IEEE*,
Bowen Zhao, Xinhan Wang, Fuhong Song, Rong Qu, *Senior Member, IEEE*, and Li Feng

***Abstract*—This paper proposes a dual-network-based feature extractor, perceptive capsule network (PCapN), for multivariate time series classification (MTSC), including a local feature network (LFN) and a global relation network (GRN). The LFN has two heads (i.e., Head_A and Head_B), each containing two squash CNN blocks and one dynamic routing block to extract the local features from the data and mine the connections among them. The GRN consists of two capsule-based transformer blocks and one dynamic routing block to capture the global patterns of each variable and correlate the useful information of multiple variables. Unfortunately, it is difficult to directly deploy PCapN on mobile devices due to its strict requirement for computing resources. So, this paper designs a lightweight capsule network (LCapN) to mimic the cumbersome PCapN. To promote knowledge transfer from PCapN to LCapN, this paper proposes a deep transformer capsule mutual (DTCM) distillation method. It is targeted and offline, using one- and two-way operations to supervise the knowledge distillation process for the dual-network-based student and teacher models. Experimental results show that the proposed PCapN and DTCM achieve excellent performance on UEA2018 datasets regarding top-1 accuracy.**

***Index Terms*—Capsule Network, Deep Learning, Data Mining, Knowledge Distillation, Multivariate Time Series Classification, Mutual Learning.**

I. Introduction

MULTIVARIATE time series (MTS) data has been utilized in various real-world applications, such as, electroencephalogram (EEG) detection [1], electromyography (EMG) decoding [2], damage anomaly analysis [3], hand movement recognition [4], abnormal event detection [5], emotion recognition [6], heart diseases analysis [7], and cognitive impairment [8] [9]. Unlike other data, such as, CIFAR10 [1] for image classification, DAQUAR [2] for visual question answering, and YFCC100M [3] for text classification, multivariate time series is a series of time-ordered data points associated with multiple time-dependent variables, each containing local and global patterns. The local patterns disclose the significant changes in the data, while the global ones unveil its overall trend. To address multivariate time series classification (MTSC) problems, one needs to extract the local and global patterns for each univariate time series (UTS) and discover the relationships among them [10], [11], [12].

MTSC belongs to time series classification (TSC). TSC algorithms can be directly used to tackle MTSC problems. These algorithms can roughly be divided into two categories, i.e., traditional and deep learning algorithms [13]. Traditional algorithms capture the features and regularizations from the input by revealing the significant differences and connections within the data. These algorithms are primarily distance-based and feature-based. Especially, combining the nearest neighbor (NN) and dynamic time warping (DTW) is one of the commonly used distance-based algorithms that measure the similarities between spatial features of data [14], such as, DTW_I [15], DTW_D [15], and DTW_A [13]. There are a variety of DTW-NN-based ensemble algorithms. For instance, Lines *et al.* [16] introduced an elastic ensemble (EE) integrating 11 1-NN-based elastic distances to explore the significant differences within the data. Bagnall *et al.* [17] put forward the collective of transformation-based ensembles (COTE) that used a number of NN-based classifiers to identify different data representations. Two representative algorithms in the literature were the hierarchical vote collective of transformation-based ensembles (HIVE-COTE) [18] and explainable-by-design ensemble method (XEM) [19].

Feature-based algorithms aim at mining the representative shapelets from time series data. For example, a bag-of-features representation framework was adopted to capture the information at different locations of sequences [20]. Hills *et al.* [21] proposed a single-scan shapelet algorithm to generate a transformed dataset. Pei *et al.* [22] presented a hidden-unit

Manuscript received XX, XX. This work was partially supported by the National Natural Science Foundation of China (No. 62172342 and No.62202392), the Natural Science Foundation of Hebei Province (No. F2022105027), the Natural Science Foundation of Sichuan Province (No. 2022NSFSC0568, No. 2022NSFSC0944, and No. 2023NSFSC0459), and the Fundamental Research Funds for the Central Universities, P. R. China (Corresponding Author: Huanlai Xing).

Z. Xiao, H. Xing, B. Zhao, X. Wang, and L. Feng are with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 610031, China, also with the Tangshan Institute of Southwest Jiaotong University, Tangshan 063000, China, and also with the Engineering Research Center of Sustainable Urban Intelligent Transportation, Ministry of Education, Chengdu 611756, China (Emails: xiao1994zw@163.com; hxx@home.swjtu.edu.cn; cn16bz@icloud.com; xhwang@my.swjtu.edu.cn; fengli@swjtu.edu.cn).

X. Xu is with the School of Computer Science and Technology, China University of Mining and Technology, Xuzhou 221166, China (Email: jhsu99@163.com).

F. Song is with the School of Information, Guizhou University of Finance and Economics, Guiyang 550025, China (fhsong@mail.gufe.edu.cn).

R. Qu is with the School of Computer Science, University of Nottingham, Nottingham NG7 2RD 455356, UK (rong.qu@nottingham.ac.uk).

[1]http://www.cs.toronto.edu/~kriz/cifar.html
[2]https://www.mpi-inf.mpg.de/
[3]http://projects.dfki.uni-kl.de/yfcc100m/

logistic model (HULM) to construct the latent structure within the data. Well-known feature-based algorithms include the learned pattern similarity [23], bag of SFA symbols (BOSS) [24], time series forest [25], TS-CHIEF [26], temporal feature fusion [27], and fuzzy cognitive map [28].

Also, there are other traditional algorithms for TSC. For instance, Baydogan *et al.* [29] presented a symbolic representation to extract the information contained in the MTS data, called SMTS. In [30], an autoregressive tree-based ensemble approach (mv-ARF) was applied to extract the non-linear dependencies and information hidden in the data. In [31], a generalized random shapelet forest algorithm was adopted for generating a series of discriminative subsequences of the data. Schäfer and Leser [32] developed an MTS classifier, WEASEL+MUSE, to capture the critical information of each variable and the relationships among the variables.

Deep learning algorithms pay attention to extracting the intrinsic connections among representations by unfolding the internal representation hierarchy of data [33]. These algorithms mainly include single-network-based and dual-network-based models [34]. To be specific, a single-network-based model mines the basic representation hierarchy of data and the significant correlations within the hierarchy by one (usually hybridized) network. For example, in [35], a multi-channel deep convolutional neural network was adopted to capture the latent features from MTS data. The off-the-shelf deep convolutional neural network (CNN) [36], InceptionTime [37], ROCKET [38], mini-ROCKET [39], deep contrastive representation learning with self-distillation [40], residual network with ensemble deep random vector functional link [41], echo state network [42], and the causal dilated convolutional encoder [11] are all single-network-based. On the contrary, a dual-network-based model is composed of two networks, one for local feature extraction and the other for global relation extraction. The local-feature extraction network, usually based on CNNs, pays attention to extracting the local features, while the global relation extraction network concentrates on exploring the relationships among the features extracted, for instance, Huang *et al.* [43] took advantage of residual networks (ResNet) and transformer-based networks for feature and relation extraction, respectively. Karim *et al.* [44] combined fully convolutional networks with long short-term memory (LSTM) networks to capture rich representations from MTS data. In [45], a time series attentional prototype network (TapNet) with multivariate time series encoding and LSTM networks was developed for MTSC. Xiao *et al.* [34] introduced a robust temporal feature network (RTFN) consisting of a temporal feature network and an LSTM-based attention network for supervised classification and unsupervised clustering. Xing *et al.* [46] designed a robust semi-supervised model with self-distillation model called SelfMatch for semi-supervised classification, where ResNet–LSTMaN including a residual network and an LSTM-based network for feature extraction.

Some useful information of an entity (object) is easily lost due to the translation invariance of neural networks, e.g., Maxpooling. To overcome the problem above, Sabour *et al.* [47] introduced a capsule network (CapNet) with dynamic routing mechanism to maintain different information of an entity, such as, location, orientation, and color. Capsule-based algorithms have been applied to a spectrum of domains [48], such as, sequential recommendation [49], machinery fault diagnosis [50], traffic analysis [51], text classification [52] and TSC [53] [54].

Addressing TSC problems by capsule-based algorithms has attracted increasingly more research attention. For example, Jayasekara *et al.* [53] introduced a 1-dimensional convolutional capsule network to extract the temporal features from the electrocardiogram data. Xiao *et al.* [54] presented a multi-process collaborative architecture based on multi-head CNN and capsule mechanism for TSC. However, CNN-based capsule algorithms are not sensitive to the global patterns of UTS, needless to say the relationships among variables of MTS. On the other hand, transformer-based capsule algorithms can relate different location information of data, which helps to extract the global patterns of UTS and relationships among multiple variables of MTS [55]. Transformer-based algorithms are widely adopted to extract the context information of sequential data and global patterns of time series data [34] [45] [56]. Some researchers hybridized the transformer- and capsule-structures for efficient global relation extraction, including cascading vanilla transformer and dynamic routing [57] [58] and embedding routing into the transformer [59]. When faced with MTSC data, the cascading structure is not sensitive to exploring the global patterns of UTS data and relationships among variables since the vanilla transformer easily breaks the rules among capsules during the transmission process; the embedding structure cannot effectively detect the relationships among variables because the embedded dynamic routing mechanism may prevent the transformer from generating a large number of capsules with rich knowledge.

To handle MTSC problems, this paper proposes a dual-network-based capsule algorithm, called perceptive capsule network (PCapN). To be specific, the PCapN primarily contains a local feature network (LFN) and a global relation network (GRN). The LFN has two heads (i.e., Head_A and Head_B), each of which cascades two squash CNN blocks and one dynamic routing block to extract the local features from MTS data and the relationships among them. The GRN consists of two capsule-based transformer blocks and one dynamic routing block to capture the global patterns of each variable and correlate the useful information of multiple variables. Nevertheless, it is impossible to deploy the dual-network-based PCapN on mobile devices for real-world applications, such as human activity recognition, ECG analysis, audio detection. This is because mobile devices usually have limited computing and storage resources, and running the PCapN on mobile devices is not affordable. It is, hence, meaningful to use a lightweight model to mimic the cumbersome PCapN in terms of feature extraction.

Hinton *et al.* [60] introduced a knowledge distillation (KD) method to transfer knowledge from a large-scale neural network to a small one, also known as the student-and-teacher model. Compared with other compression and acceleration techniques, such as parameter pruning/sharing and low-rank factorization, KD enables a student model to effectively mimic the knowledge of a teacher model and regularizes the student and teacher models to promote the mutual flow of knowledge between them [61] [62]. In traditional KD, a teacher's knowledge was allowed to flow to a student. Recently, some research findings suggested the knowledge learned by a student also promoted its teacher's knowledge [61] [63] [64]. In other words, KD can be viewed as a mutual learning method (also called two-way knowledge transfer). Yao *et al.* [64] designed a dense cross-layer mutual (DCM) distillation to promote the dense flow of knowledge among different layers between the student and teacher models and effectively reduce the loss of useful features during the distillation process.

DCM achieved decent performance with single-network-based student and teacher models. Its distillation provides adequate interaction among the layers of these models, ensuring rich knowledge flowing from teacher to student. Currently, DCM is regarded as one of the state-of-the-art methods for mutual learning. However, DCM is not directly applicable to dual-network-based student and teacher models since a dual-network-based model consists of two networks with different objectives, e.g., one for local feature extraction and the other for global relation extraction. In other words, the two networks concentrate on different feature types. If DCM is applied to dual-network-based student and teacher models, the dense interactive mutual learning between student and teacher could generate some incorrect or unnecessary information that disturbs efficient knowledge transfer between them, possibly losing the latent information accumulated during the distillation process. To the best of our knowledge, efficient mutual learning for dual-network-based student and teacher models has not received sufficient research attention.

To tackle the problem above, we present a deep transformer capsule mutual (DTCM) distillation to supervise the knowledge transfer process from the proposed PCapN to a lightweight capsule network (LCapN). The PCapN and LCapN are the teacher and student models, respectively. The LCapN's architecture is similar to the PCapN's to ensure smooth knowledge flow between them. The LCapN includes a local feature network (LFN) and a global relation network (GRN). The LFN consists of two lightweight squash CNN blocks and one dynamic routing block, while the GRN has two lightweight capsule-based transformer blocks and one dynamic routing block. In terms of parameter size, the LCapN is three times smaller than the PCapN.

Our main contributions are summarized below.

- To address MTSC problems, we design PCapN to extract rich representations from data. The PCapN mainly consists of a local feature network and a global relation network. Based on squash CNN block and dynamic routing, the local feature network extracts local features. The global relation network is primarily composed of two capsule-based transformer blocks and one dynamic routing block, responsible for capturing the global patterns for all UTS in the MTS data and discovering the relationships among them.
- To efficiently transfer knowledge from the PCapN to the LCapN, we design the DTCM distillation. Unlike the DCM, which is suitable for single-network-based models, our DTCM is a targeted and offline distillation method for dual-network-based student and teacher models, providing adequate knowledge transfer between the PCapN and LCapN. To our best, DTCM is the first mutual learning method for dual-network-based distillation models.
- Experimental results show that the PCapN overweighs a large number of existing MTSC algorithms on 10 UEA2018 datasets in terms of the 'win'/'tie'/'lose' measure and AVG_rank, both based on the top-1 accuracy. Compared with 19 state-of-the-art KD methods, DTCM achieves excellent performance on 13 out of 30 UEA2018 datasets concerning the mean accuracy, 'win'/'tie'/'lose' measure, and AVG_rank.

The rest of the paper is organized below. Section II reviews capsule-based and KD algorithms. Section III first introduces the proposed PCapN and LCapN and their key components and then describes the DTCM method that promotes sufficient knowledge transfer between the PCapN and LCapN. Section IV provides and analyzes the experimental results, and the conclusion draws in Section V.

II. Related Work

This section reviews capsule-based and KD algorithms.

A. Capsule Network

The "capsule" concept was originated from parse trees for transforming auto-encoders [65]. A capsule consists of multiple neurons, each of which denotes an entity or part of an entity in a parse tree. The output of a capsule is the instantiation parameters representing properties of a specific type of entity, with the capsule's length as the entity's probability. To further relate the potential information among capsules at different locations, Sabour *et al.* [47] introduced a capsule network with dynamic routing mechanism. Since then, increasingly more capsule-based algorithms have been developed to tackle various real-world applications [48]. CNN- and transformer-based capsule algorithms are two most representative capsule-based algorithms. CNN-based capsule algorithms often utilize CNNs and dynamic routing to explore the local features from the input data and the relationships among these features. For example, Huang *et al.* [50] presented a robust weight-shared capsule network using 1-dimensional CNNs and multi-stacked weight-shared capsules for intelligent machinery fault diagnosis. Yao *et al.* [51] introduced a deep learning aided capsule network based on CNNs and dynamic routing for traffic classification. Peng *et al.* [52] applied attentional graph capsule embedding recurrent CNNs to multi-label text classification, where CNNs were used to learn the semantic features of word matrices. Zhao *et al.* [66] adopted a 3D point-capsule network consisting of multilayer perceptrons, Maxpooling, and dynamic routing to process sparse 3D point clouds. Yang *et al.* [67] presented a hierarchical graph capsule network that used graph neural networks to extract hierarchical representations from graph data. A multi-process collaborative architecture based on multi-head CNNs and capsule mechanism was designed to address TSC problems [54].

Transformer-based capsule algorithms are responsible for extracting the context information of sequential data and the global patterns of time series data efficiently [34] [45], thanks to its transformer that relates the information at different locations [55]. Hybridizing transformer- and capsule-structures is one of the most popular research streams. For example, in [57] [58], some cascading structures were proposed for stock movements prediction and tweet act classification, where vanilla transformer and dynamic routing were stacked together. An embedding structure was put forward for machine translation, where routing mechanism was embedded into the transformer's attention part [59].

On the one hand, the cascading structure is not appropriate to handle MTSC data because this structure cannot effectively detect the global patterns of UTS data and the relationships among variables. In the vanilla transformer, its vanilla attention module blocks the interaction among capsules during the knowledge transfer process, causing that the transformer cannot provide sufficient capsules that carry rich knowledge for the dynamic routing to explore the relationships among them. On the other hand, the embedding structure cannot effectively sense the relationships among variables because the embedded dynamic routing mechanism may interfere with the process in the transformer that generates sufficient capsules with high-level features. This is why we design the GRN for global relation extraction in the PCapN. The GRN consists of two capsule-based transformer blocks and one dynamic routing. These transformers can generate plenty of capsules full of global patterns of all UTS in the MTS data, because the attention modules enhance the interaction rules to capture the potential correlation among the capsules during knowledge transfer. The dynamic routing relates the previously generated capsules to discover the intrinsic relationships among the variables in the MTS data.

B. Knowledge Distillation

KD is a knowledge transfer process from a complex network to a lightweight network. Currently, KD algorithms can be roughly divided into four categories, i.e., response-based, feature-based, relation-based and other KD algorithms [60]. For an arbitrary teacher model, if it is pre-trained, it distills its student(s) online; otherwise, it distills its student(s) offline.

1) Response-based KD: Given a student and teacher model, response-based KD algorithms usually make the output (i.e., logits) of the student model mimic that of the teacher model directly [60]. These algorithms are widely used in a variety of real-world applications. For instance, Chen *et al.* [68] presented a KD framework for fast multi-class object detection. Zhang *et al.* [69] proposed a fast pose distillation (FPD) model for knowledge transfer and extraction. Meng *et al.* [70] adopted a conditional teacher-student learning to intelligently select what knowledge to learn from a teacher model. Besides, some researchers dedicated their efforts to interpret "dark knowledge" generated by the response-based KD, for example, Müller *et al.* [71] regularized the student and teacher models using soft targets.

2) Feature-based KD: Feature-based KD algorithms transfer knowledge from the teacher to student models using the feature maps of intermediate layers. Romero *et al.* [72] expanded the teacher-student knowledge transfer from the output-to-output to intermediate-to-intermediate layers. After this pioneering work, increasingly more research interests have been dedicated to feature-based KD. Liu *et al.* [73] introduced a knowledge representing (KR) method that modeled the distribution of parameters. Kim *et al.* [74] used a factor-transfer-based method to offer efficient flow of knowledge from teacher to student. Zhou *et al.* [75] designed a rocket launching framework using gradient block to improve the performance of light and booster nets. Guan *et al.* [76] put forward a differentiable feature aggregation (DFA) search method to search for useful knowledge between student and teacher.

3) Relation-based KD: Relation-based KD ones are responsible for extracting the relationships among layers in the student and teacher models. Tung *et al.* [77] proposed a similarity-preserving (SP) KD to supervise knowledge transfer via activation similarity matrices. Yu *et al.* [78] used a metric learning KD (MLKD) to ensure that the student model efficiently computes image embeddings. Liu *et al.* [79] modeled the feature space transformation across layers of student and teacher models using instance relationship graph (IRG). Chen *et al.* [80] used the feature-embedding-based learning student networks (LSN) to propagate teacher-to-student knowledge.

4) Other KD: Yuan *et al.* [62] introduced a teacher-free knowledge distillation (Tf-KD) framework, where student learned from either itself or a manually designed regularization distribution. To further understand the "dark knowledge" in the KD process, Zhang *et al.* [63] assumed that the knowledge flow between student and teacher could be bidirectional, i.e., both student and teacher could learn from each other. Therefore, they proposed a deep mutual learning (DML) framework, also called the two-way KD, to facilitate the knowledge transfer between student and teacher. As DML was based on layer-to-layer monitoring only, it was impossible for high-level representations to have a direct yet positive impact on low-level representations, easily resulting in the loss of useful information during distillation. To this end, Yao *et al.* [64] designed a dense cross-layer mutual (DCM) distillation to promote the dense flow of knowledge among different layers between student and teacher and alleviate feature loss during distillation.

DCM, one of the well-known methods for mutual learning, is suitable for distillation between single-network-based student and teacher models but not between dual-network-based student and teacher models. A dual-network-based model consists of two networks emphasizing extracting different feature types. If not appropriately designed, DCM could damage the previously extracted latent information, causing inefficient distillation supervision. This motivates us to develop DTCM, an appropriate targeted and offline distillation method for dual-network-based student and teacher models, providing adequate knowledge transfer between the PCapN and LCapN. To the best of our knowledge, DTCM is the first mutual learning method for dual-network-based distillation models.

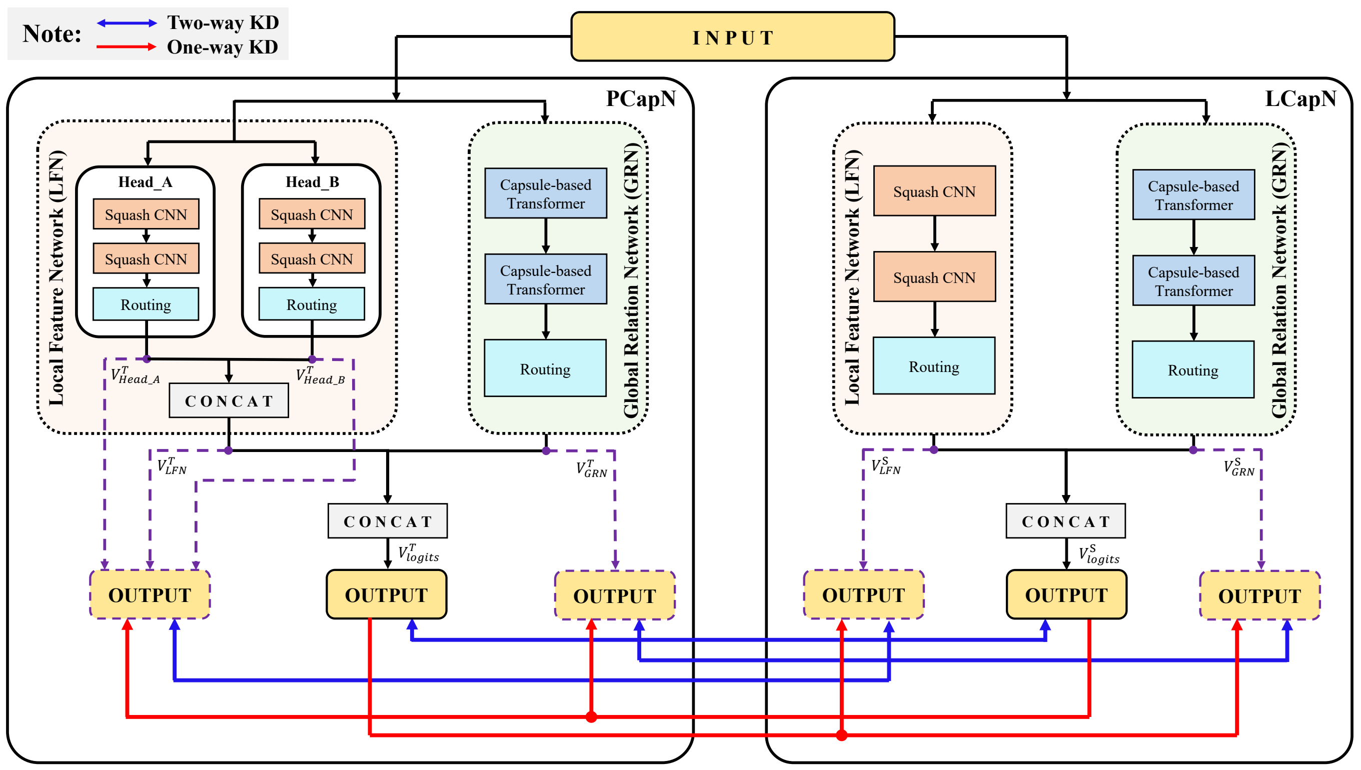


Fig. 1. Structure of DTCM. Note: $V^{\mathcal{T}}_{Head\_A}$, $V^{\mathcal{T}}_{Head\_B}$, $V^{\mathcal{T}}_{LFN}$, $V^{\mathcal{T}}_{GRN}$ are the outputs of Head_A, Head_B, LFN, and GRN in the PCapN, respectively; $V^{\mathcal{T}}_{logits}$ denotes the output of PCapN; $V^{\mathcal{S}}_{LFN}$ and $V^{\mathcal{S}}_{GRN}$ represent the outputs of LFN and GRN in the LCapN, respectively; $V^{\mathcal{S}}_{logits}$ is the output of LCapN.

III. Teacher and Student Models with DTCM

This section first introduces the PCapN's and LCapN's structures and their key components, including the squash CNN block, dynamic routing block, and capsule-based transformer block. Then, it describes the DTCM method that promotes efficient knowledge transfer between the PCapN and LCapN.

A. Overview

The PCapN, as the teacher model, primarily consists of a local feature network (LFN) and a global relation network (GRN), as shown in Fig. 1. The LFN includes two heads, namely, Head_A and Head_B, each of which stacks two squash CNN blocks and one dynamic routing block together to extract the local features from MTS data and the relationships among these features. The GRN consists of two capsule-based transformer blocks and one dynamic routing block to capture each variable's global patterns and correlate the useful information of multiple variables. On the other hand, LCapN is also composed of an LFN and a GRN. Its LFN consists of two lightweight squash CNN blocks and one dynamic routing block, while its GRN cascades two lightweight capsule-based transformer blocks and one dynamic routing block.

This paper presents the DTCM method to promote knowledge transfer between the PCapN and LCapN. This method uses two-way and one-way KD operations to realize efficient knowledge flow between the student and teacher models, shown in Fig. 1. On the one hand, by using two-way KD operations, the outputs of Head_A, Head_B, and LFN in the PCapN, interact with the output of LFN in the LCapN; the output of GRN in the PCapN and that of GRN in the LCapN interact with each other; the output of PCapN and that of LCapN interact with each other. These operations promote the two-way knowledge transfer between the PCapN and LCapN. On the other hand, by using one-way KD operations, the PCapN's output guides the outputs of LFN and GRN in the LCapN while the LCapN's output guides the outputs of Head_A, Head_B, LFN, and GRN in the PCapN.

B. Squash CNN Block

There are two squash CNN blocks in the PCapN. The first block extracts the basic local features from the input. The second one mines the high-level features from the features extracted by the first block. The two CNN blocks generate sufficient capsules with rich features which are then fed to the dynamic routing block that mines the connections among these capsules. A squash CNN block includes a 1-dimensional CNN module, a batch normalization module, and a squashing non-linear activation function, defined as:

$$V_{squash\_cnn} = f_{squash}(f_{BN}(f_{conv}(\mathcal{X}))) \tag{1}$$

where, V_{squash_cnn} and $\mathcal{X}$ are the output and input of the squash CNN block, respectively. f_{squash}, f_{BN}, and f_{conv} represent the squashing non-linear activation, batch normalization, and CNN functions, respectively.

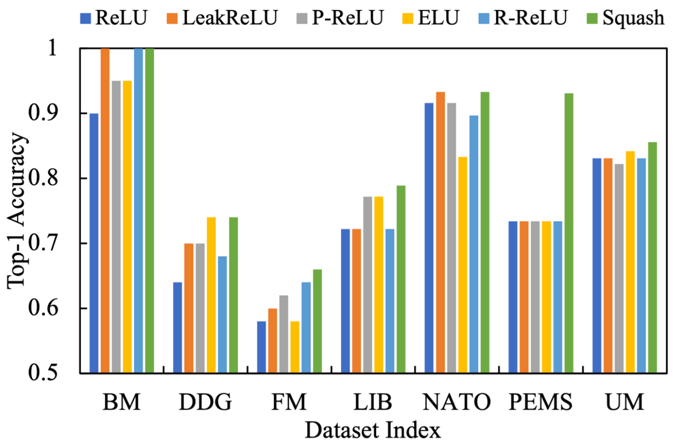


Fig. 2. Top-1 accuracy values of PCapN with various activation functions on 7 popular UEA datasets, including BasicMotions (BM), DuckDuck-Geese (DDG), FingerMovements (FM), Libras (LIB), NATOPS (NATO), PEMSF (PEMS), and UWaveGuestureLibrary (UM). Abbreviations: ReLU–rectified linear unit, LeakReLU–leaky ReLU, P-ReLU–Parametric ReLU, ELU–exponential linear unit, R-ReLU–randomized leaky ReLU.

A CNN module is used to extract the local features from the data [33], as defined in Eq. (2).

$$f_{conv}(\mathcal{X}_{conv}) = \mathcal{X}_{conv} \otimes W_{conv} + b_{conv} \tag{2}$$

where, $\mathcal{X}_{conv}$, W_{conv} and b_{conv} are the input, weight and bias metrics of the CNN, respectively. $\otimes$ denotes the convolutional computation operation.

Following [34] [54], we use batch normalization to eliminate the internal covariate shift, ensuring a relatively fast training process. Also, it enhances the local-feature extraction ability of LFN on the MTS data. Following [47] [48] [49] [50], we adopt the squashing activation function to activate neurons. f_{squash} is defined in Eq. (3).

$$f_{sqaush}(\mathcal{X}_{squash}) = \frac{||\mathcal{X}_{squash}||^2}{1 + ||\mathcal{X}_{squash}||^2} \frac{\mathcal{X}_{squash}}{||\mathcal{X}_{squash}||} \tag{3}$$

where, $\mathcal{X}_{squash} = \{\mathcal{X}^1_{squash}, \mathcal{X}^2_{squash}, ..., \mathcal{X}^{n_{squash}}_{squash}\}$ is the input of squashing activation function, and n_{squash} is the size of the input. $||\mathcal{X}_{squash}|| = \sqrt{(\mathcal{X}^1_{squash})^2 + (\mathcal{X}^2_{squash})^2 + ... + (\mathcal{X}^{n_{squash}}_{squash})^2}$ outputs the length of $\mathcal{X}_{squash}$, ensuring that $\mathcal{X}_{squash}$ can derive its length regardless of whether it is positive or not.

Distinct from conventional neural network architectures, CapsNet transmits data in the form of vectors rather than scalars [47]. As a result, employing the squashing activation function, specifically tailored for vectors, enhances the feature extraction capabilities of our PCapN. We study the influence of different activation functions on PCapN on 7 popular UEA datasets in Fig. 2. The experimental results demonstrate that compared with widely used activation functions, e.g., the rectified linear unit (ReLU) and the leaky ReLU, the squashing activation function empowers our PCapN to extract more high-quality semantic information from the data.

Algorithm 1 Dynamic Routing Mechanism

```
 1: procedure ROUTING($s^{inp}_j, r$)   ▷ $s^{inp}_j$ is the input vector of capsule $j$, and $r$ denotes the number of iterations.
 2: Initialize weight matrix $W_{ij}$;
 3: Set $\hat{v}_{j|i} = W_{ij} s^{inp}_j$ and $b_{ij} = 0$;
 4:     for $r$ iterations do
 5:         Calculate $k_{ij}$ by using Eq. (5);
 6:         Calculate $\hat{v}_{j|i}$ and $s_j$ by using Eq. (4);
 7:         Calculate $v_j = f_{squash}(s_j)$ by using Eq. (3);
 8:         Calculate $b_{ij}$ by using Eq. (6);
 9:     end for
10: return $v_j$;
11: end procedure
```

C. Dynamic Routing Block

As [34] [52] [53] [54] suggested, we use dynamic routing to promote the interaction among capsules, helping to mine the relationships among them. For an arbitrary capsule j, its input s_j is computed as:

$$s_j = \sum_i k_{ij} \hat{v}_{j|i}, \quad \hat{v}_{j|i} = W_{ij} v_i \tag{4}$$

where, $\hat{v}_{j|i}$ is the prediction vector calculated by multiplying the output of capsule i in the previous layer (i.e., v_i) by the weight matrix W_{ij}. The coupling coefficient between all capsules in the current layer and capsule i in the previous layer, k_{ij}, is obtained by a softmax function through an iterative routing process [53], [54], [55]. k_{ij} is defined in Eq. (5).

$$k_{ij} = f_{softmax}(b_{ij}) \tag{5}$$

where, $f_{softmax}$ is a commonly used function to compute the possibilities of a given matrix, and b_{ij} is the logits representing the log prior probabilities that capsule i couples to capsule j.

b_{ij} is refined iteratively by measuring the "agreement" between the current output v_j and the prediction $\hat{v}_{j|i}$ obtained by capsule i from the previous layer, defined in Eq. (6).

$$b_{ij} = b_{ij} + v_j \cdot \hat{v}_{j|i} \tag{6}$$

where, v_j is the output of capsule j and calculated by "squashing" its input, s_j, in the current layer. The pseudo-code of dynamic routing is given in Alg. 1.

D. Capsule-based Transformer Block

In the PCapN, two capsule-based transformer blocks are cascaded to generate sufficient number of capsules with adequate global patterns of all UTS in the MTS data. The first block extracts the basic relationships from the MTS data while the second one relates these basic relationships at different locations to discover the potential representations hidden in the data. In this way, plenty of capsules with rich relationship information are generated and passed to the dynamic routing block for further processing.

A capsule-based transformer block extracts the internal relationships and rules among those representations, by relating the representations at different locations of the MTS data, as shown in Fig. 3. The multi-head capsule-based attention

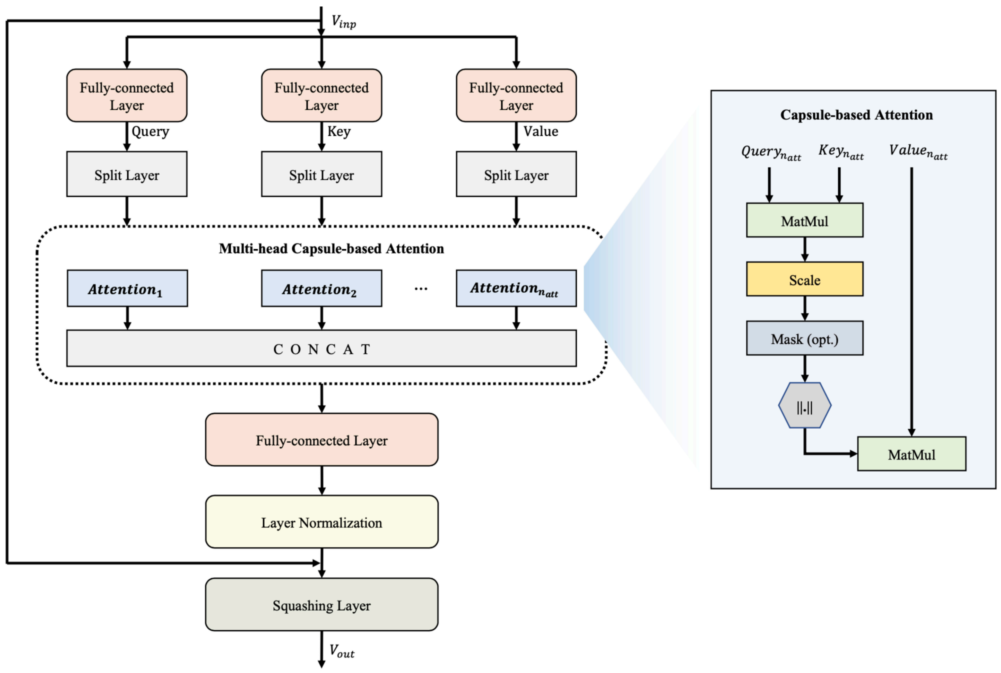


Fig. 3. Diagram of the capsule-based transformer. Note: V_{inp} and V_{out} denote the input and output vectors of the capsule-based transformer block, respectively; n_{att} is the number of capsule-based attention modules; ||.|| outputs the length of a given vector; 'MatMul' is a matrix multiplication operation.

network is the block's core and consists of n_{att} capsule-based attention modules. For an arbitrary capsule-based attention module, i.e., $Attention_i$, it maps a query, $Query_i \in \mathbb{R}^{n\times L\times d}$, and a set of key-values, Key_i-$Value_i \in \mathbb{R}^{n\times L\times d}$, to an output, V_{att}^i, where n is the the number of input samples, L represents the length of the MST data, and d indicates the number of univariate signals or variables within the MST data. Unlike the vanilla attention module [57][58], the capsule-based attention module considers the interaction rules among capsules, e.g., the length of each capsule's vector represents the probability that the capsule's entity exists, which is beneficial for extracting the intrinsic connections between a query and its corresponding key-value pairs. V_{att}^i is defined in Eq. (7).

$$V_{att}^i = ||\frac{Query_i \cdot Key_i^T}{\sqrt{d}}|| \cdot Value_i \tag{7}$$

where, Key_i^T is the transpose of Key_i.

Let V_{mul} denote the output of the multi-head capsule-based attention network. V_{mul} combines the outputs of the n_{att} capsule-based attention modules through the CONCAT function, f_{concat}, to provide adequate global features. V_{mul} is defined in Eq. (8).

$$V_{mul} = f_{concat}([V_{att}^1, V_{att}^2, ..., V_{att}^{n_{att}}]) \tag{8}$$

We add a residual operation to compensate for the loss of useful information during the feature transmission process. The output of the capsule-based transformer block, V_{out}, is written as:

$$V_{out} = f_{squash}(f_{LN}(f_{connect}(V_{mul})) + V_{inp}) \tag{9}$$

where, f_{LN} presents the layer normalization function, and V_{inp} is the input of the capsule-based transformer block. Like other transformer structures [55] [57] [58], we use layer normalization to accelerate the model's convergence and avoid overfitting during training.

E. Deep Transformer Capsule Mutual Distillation

The DTCM is a targeted distillation method for dual-network-based student and teacher models, promoting efficient knowledge flow between them. Let $V_{Head\_A}^{\mathcal{T}}$, $V_{Head\_B}^{\mathcal{T}}$, $V_{LFN}^{\mathcal{T}}$, $V_{GRN}^{\mathcal{T}}$ be the outputs of Head_A, Head_B, LFN, and GRN in the PCapN, respectively. Let $V_{logits}^{\mathcal{T}}$ denote the output of PCapN. Let $V_{LFN}^{\mathcal{S}}$ and $V_{GRN}^{\mathcal{S}}$ represent the outputs of LFN and GRN in the LCapN, respectively. Let $V_{logits}^{\mathcal{S}}$ denote the output of LCapN.

The loss function of PCapN, $\mathcal{L}^{\mathcal{T}}$, consists of four components, i.e., margin loss, $\mathcal{L}_{margin}^{\mathcal{T}}$, LFN loss, $\mathcal{L}_{LFN}^{\mathcal{T}}$, GRN loss, $\mathcal{L}_{GRN}^{\mathcal{T}}$, and logits loss, $\mathcal{L}_{logits}^{\mathcal{T}}$, as defined in Eq. (10):

$$\mathcal{L}^{\mathcal{T}} = \mathcal{L}_{margin}^{\mathcal{T}} + \alpha^{\mathcal{T}}\mathcal{L}_{LFN}^{\mathcal{T}} + \beta^{\mathcal{T}}\mathcal{L}_{GRN}^{\mathcal{T}} + \gamma^{\mathcal{T}}\mathcal{L}_{logits}^{\mathcal{T}} \tag{10}$$

where, $\alpha^{\mathcal{T}}$, $\beta^{\mathcal{T}}$, and $\gamma^{\mathcal{T}}$ are three constant coefficients. As [64] suggested, we set $\alpha^{\mathcal{T}} = 1$, $\beta^{\mathcal{T}} = 1$, and $\gamma^{\mathcal{T}} = 1$. We adopt the commonly used margin loss function for capsule-based algorithms [47] [48] [53] [54] as $\mathcal{L}_{margin}^{\mathcal{T}}$.

$\mathcal{L}_{LFN}^{\mathcal{T}}$, $\mathcal{L}_{GRN}^{\mathcal{T}}$, and $\mathcal{L}_{logits}^{\mathcal{T}}$ are defined as:

$$\begin{aligned}\mathcal{L}_{LFN}^{\mathcal{T}} =& \mathcal{L}_{KD}(V_{Head\_A}^{\mathcal{T}}, V_{LFN}^{\mathcal{S}}) + \mathcal{L}_{KD}(V_{Head\_B}^{\mathcal{T}}, V_{LFN}^{\mathcal{S}}) \\ &+ \mathcal{L}_{KD}(V_{LFN}^{\mathcal{T}}, V_{LFN}^{\mathcal{S}})\end{aligned} \tag{11}$$

$$\mathcal{L}_{GRN}^{\mathcal{T}} = \mathcal{L}_{KD}(V_{GRN}^{\mathcal{T}}, V_{GRN}^{\mathcal{S}}) \tag{12}$$

$$\begin{aligned}\mathcal{L}_{logits}^{\mathcal{T}} =& \mathcal{L}_{KD}(V_{logits}^{\mathcal{T}}, V_{LFN}^{\mathcal{S}}) + \mathcal{L}_{KD}(V_{logits}^{\mathcal{T}}, V_{GRN}^{\mathcal{S}}) \\ &+ \mathcal{L}_{KD}(V_{logits}^{\mathcal{T}}, V_{logits}^{\mathcal{S}})\end{aligned} \tag{13}$$

TABLE I
DETAILS OF 30 MULTIVARIATE TIME SERIES DATASETS. ABBREVIATIONS: HAR - HUMAN ACTIVITY RECOGNITION, AS - AUDIO SPECTRA, EEG - ELECTROENCEPHALOGRAM, MEG - MAGNETOENCEPHALOGRAPHY, ECG - ELECTROCARDIOGRAM.

Index	DatasetName	SeriesLength	NumDimensions	NumClasses	TrainSize	TestSize	Type
AWR	ArticularyWordRecognition	144	9	25	275	300	Motion
AF	AtrialFibrillation	640	2	3	15	15	ECG
BM	BasicMotions	100	6	4	40	40	HAR
CT	CharacterTrajectories	182	3	20	1422	1436	Motion
CK	Cricket	1197	6	12	108	72	HAR
DDG	DuckDuckGeese	270	1345	5	50	50	AS
EW	EigenWorms	17984	6	5	128	131	Motion
EP	Epilepsy	206	3	4	137	138	HAR
EC	EthanolConcentration	1751	3	4	261	263	HAR
ER	ERing	65	4	6	30	270	Other
FD	FaceDetection	62	144	2	5890	3524	EEG/MEG
FM	FingerMovements	50	28	2	316	100	EEG/MEG
HMD	HandMovementDirection	400	10	4	160	74	EEG/MEG
HW	Handwriting	152	3	26	150	850	HAR
HB	Heartbeat	405	61	2	204	205	AS
IW	InsectWingbeat	30	200	10	30000	20000	AS
JV	JapaneseVowels	29	12	9	270	370	AS
LIB	Libras	45	2	15	180	180	HAR
LSST	LSST	36	6	14	2459	2466	Other
MI	MotorImagery	3000	64	2	278	100	EEG/MEG
NATO	NATOPS	51	24	6	180	180	HAR
PD	PenDigits	8	2	10	7494	3498	EEG/MEG
PEMS	PEMS-SF	144	963	7	267	173	EEG/MEG
PS	Phoneme	217	11	39	3315	3353	AS
RS	RacketSports	30	6	4	151	152	HAR
SRS1	SelfRegulationSCP1	896	6	2	268	293	EEG/MEG
SRS2	SelfRegulationSCP2	1152	7	2	200	180	EEG/MEG
SAD	SpokenArabicDigits	93	13	10	6599	2199	AS
SWJ	StandWalkJump	2500	4	3	12	15	ECG
UW	UWaveGestureLibrary	315	3	8	120	320	HAR

where, $\mathcal{L}_{KD}$ is a knowledge distillation loss function that measures the average difference between the outputs of the student and teacher models. To be specific, $\mathcal{L}_{KD}$ is based on the L_2 loss function, defined as:

$$\mathcal{L}_{KD}(V^{\mathcal{T}}, V^{\mathcal{S}}) = \frac{1}{n_{kd}} \sum_{i=1}^{n_{kd}} (f_{pre}(v_i^{\mathcal{T}}) - f_{pre}(v_i^{\mathcal{S}}))^2 \tag{14}$$

where, $V^{\mathcal{T}} = \{v_i^{\mathcal{T}}\}_{i=1}^{n_{kd}}$ and $V^{\mathcal{S}} = \{v_i^{\mathcal{S}}\}_{i=1}^{n_{kd}}$ represent the outputs of the teacher and student models. n_{kd} is the length of $V^{\mathcal{T}}$. f_{pre} is a prediction function to calculate the probability of a vector, as defined in Eq. (15).

$$f_{pre}(v_{pre\_in}) = ||v_{pre\_in}/\Gamma|| \tag{15}$$

where, Γ is a temperature coefficient for producing a soft probability distribution over classes. Following [64], we set $\Gamma = 1$ (more details can be found in Section 4.4.1).

The loss function of LCapN, $\mathcal{L}^{\mathcal{S}}$, consists of margin loss, $\mathcal{L}^{\mathcal{S}}_{margin}$, LFN loss, $\mathcal{L}^{\mathcal{S}}_{LFN}$, GRN loss, $\mathcal{L}^{\mathcal{S}}_{GRN}$, and logits loss, $\mathcal{L}^{\mathcal{S}}_{logits}$, defined as:

$$\mathcal{L}^{\mathcal{S}} = \mathcal{L}^{\mathcal{S}}_{margin} + \alpha^{\mathcal{S}} \mathcal{L}^{\mathcal{S}}_{LFN} + \beta^{\mathcal{S}} \mathcal{L}^{\mathcal{S}}_{GRN} + \gamma^{\mathcal{S}} \mathcal{L}^{\mathcal{S}}_{logits} \tag{16}$$

where, $\alpha^{\mathcal{S}}$, $\beta^{\mathcal{S}}$, and $\gamma^{\mathcal{S}}$ are three constant coefficients for the LcapN. Following [64], we set $\alpha^{\mathcal{S}} = 1$, $\beta^{\mathcal{S}} = 1$, and $\gamma^{\mathcal{S}} = 1$ in this paper.

We adopt the same margin loss function for capsule-based algorithms [47] [48] [53] [54] as $\mathcal{L}^{\mathcal{S}}_{margin}$. $\mathcal{L}^{\mathcal{S}}_{LFN}$, $\mathcal{L}^{\mathcal{S}}_{GRN}$, and $\mathcal{L}^{\mathcal{S}}_{logits}$ are defined as:

$$\begin{aligned}\mathcal{L}^{\mathcal{S}}_{LFN} =& \mathcal{L}_{KD}(V^{\mathcal{S}}_{LFN}, V^{\mathcal{T}}_{Head\_A}) + \mathcal{L}_{KD}(V^{\mathcal{S}}_{LFN}, V^{\mathcal{T}}_{Head\_B}) \\ &+ \mathcal{L}_{KD}(V^{\mathcal{S}}_{LFN}, V^{\mathcal{T}}_{LFN})\end{aligned} \tag{17}$$

$$\mathcal{L}^{\mathcal{S}}_{GRN} = \mathcal{L}_{KD}(V^{\mathcal{S}}_{GRN}, V^{\mathcal{T}}_{GRN}) \tag{18}$$

$$\begin{aligned}\mathcal{L}^{\mathcal{S}}_{logits} =& \mathcal{L}_{KD}(V^{\mathcal{S}}_{logits}, V^{\mathcal{T}}_{Head\_A}) + \mathcal{L}_{KD}(V^{\mathcal{S}}_{logits}, V^{\mathcal{T}}_{Head\_B}) \\ &+ \mathcal{L}_{KD}(V^{\mathcal{S}}_{logits}, V^{\mathcal{T}}_{LFN}) + \mathcal{L}_{KD}(V^{\mathcal{S}}_{logits}, V^{\mathcal{T}}_{GRN}) \\ &+ \mathcal{L}_{KD}(V^{\mathcal{S}}_{logits}, V^{\mathcal{T}}_{logits})\end{aligned} \tag{19}$$

Similar to other mutual learning algorithms [63] [64], we adopt gradient descent to jointly optimize the parameters of the student and teacher models. Let $\theta_i^{\mathcal{T}}$ and $\theta_i^{\mathcal{S}}$ denote the parameters of the teacher and student in the i-th training epoch, respectively. $\theta_i^{\mathcal{T}}$ and $\theta_i^{\mathcal{S}}$ are defined in Eq. (20).

$$\begin{aligned}\theta_i^{\mathcal{T}} &= \theta_{i-1}^{\mathcal{T}} - \eta^{\mathcal{T}} \nabla_{\theta_{i-1}^{\mathcal{T}}} \mathcal{L}(\theta_{i-1}^{\mathcal{T}}), \\ \theta_i^{\mathcal{S}} &= \theta_{i-1}^{\mathcal{S}} - \eta^{\mathcal{S}} \nabla_{\theta_{i-1}^{\mathcal{S}}} \mathcal{L}(\theta_{i-1}^{\mathcal{S}})\end{aligned} \tag{20}$$

where, $\nabla_{\theta_{i-1}^{\mathcal{T}}}$ and $\nabla_{\theta_{i-1}^{\mathcal{S}}}$ are the gradients of the teacher and student in the (i-1)-th training epoch, respectively. $\eta^{\mathcal{T}}$ and $\eta^{\mathcal{S}}$ represent the teacher's and student's learning rates, respectively. The DTCM's pseudo-code is shown in Alg. 2.

IV. EXPERIMENTS AND ANALYSIS

This section first introduces the experimental setup and performance metrics. After that, it thoroughly evaluates the

Algorithm 2 DTCM

```
Input: $D = \{D_{train}, D_{val}, D_{test}\}$;    ▷ $D_{train}$, $D_{val}$, and $D_{test}$ are the training, validation, and testing data, respectively.
Output: $Y^{\mathcal{T}}$ and $Y^{\mathcal{S}}$;    ▷ $Y^{\mathcal{T}}$ and $Y^{\mathcal{S}}$ are the prediction results of PCapN and LCapN, respectively.
 1: Initialize the PCapN's and LCapN's parameters, $\theta_0^{\mathcal{T}}$ and $\theta_0^{\mathcal{S}}$;
 2: //Training on training and validation data
 3: for $i = 1$ to $M$ do    ▷ $M$ is the number of epochs.
 4:     Feedforward $D_{train}$ into the PCapN and LCapN;
 5:     Compute $\mathcal{L}^{\mathcal{T}}$ and $\mathcal{L}^{\mathcal{S}}$ by using Eqs. (10)(16);
 6:     Update $\theta_i^{\mathcal{T}}$ and $\theta_i^{\mathcal{S}}$ by using Eq. (20);
 7:     if $i\%2 == 0$ then
 8:         Validate the PCapN and LCapN using $D_{val}$;
 9:     end if
10: end for
11: //Testing
12: Use the trained PCapN to predict $Y^{\mathcal{T}}$ of $D_{test}$;
13: Use the trained LCapN to predict $Y^{\mathcal{S}}$ of $D_{test}$.
```

TABLE II
FLOPs, PARAMETERS, AND INFERENCE TIME RESULTS WITH THE PCapN AND LCapN ON DIFFERENT TESTING DATASETS. ABBREVIATION: FLOPs–FLOATING POINT OPERATIONS.

Dataset Index	PCapN				LCapN			
	FLOPs (M)	Parameters (M)	GPU (s)	CPU (s)	FLOPs (M)	Parameters (M)	GPU (s)	CPU (s)
AWR	22.283	11.143	4.560	11.277	6.957	3.479	3.765	4.330
AF	11.927	5.965	3.553	2.558	3.721	1.861	3.321	1.800
BM	2.606	1.304	3.802	1.338	0.808	0.405	3.410	1.006
CT	22.624	11.313	7.955	49.719	7.041	3.521	5.683	18.141
CR	88.513	44.258	5.314	23.741	27.633	13.817	3.915	16.074
DDG	14.473	7.238	4.813	4.636	4.435	2.218	3.390	2.377
EP	5.222	2.612	3.654	2.940	1.621	0.811	3.431	1.793
EC	43.180	21.591	6.624	114.801	13.485	6.743	5.900	101.262
ER	2.591	1.297	3.705	2.272	0.793	0.397	3.650	1.364
FD	1.541	0.772	10.067	9.337	0.465	0.233	8.637	5.625
FM	0.874	0.438	3.921	1.230	0.263	0.132	3.305	0.958
HMD	9.997	5.000	4.709	4.000	3.117	1.559	4.354	2.748
HW	24.419	12.211	5.789	30.560	7.625	3.813	4.518	11.153
HB	5.390	2.696	3.865	6.954	1.671	0.836	3.646	5.416
IW	2.432	1.217	52.337	60.794	0.731	0.366	41.567	28.353
JV	1.863	0.933	4.953	2.064	0.560	0.280	4.775	1.282
LIB	4.418	2.210	3.683	2.176	1.348	0.675	3.429	1.317
LSST	3.246	1.625	7.339	11.847	1.008	0.505	6.753	5.258
MI	37.272	18.637	7.412	45.329	11.638	5.819	6.540	38.658
NATO	2.128	1.065	4.228	1.657	0.654	0.328	3.440	1.149
PD	0.623	0.313	9.128	4.222	0.188	0.095	8.262	3.258
PEMS	10.630	5.316	5.235	6.291	3.262	1.631	3.819	3.145
PS	52.532	26.267	17.307	250.919	16.342	8.172	9.683	82.594
RS	0.910	0.457	3.576	1.475	0.273	0.137	3.383	1.043
SRS1	11.158	5.580	4.654	33.330	3.480	1.741	4.277	30.533
SRS2	14.309	7.156	4.791	33.190	4.465	2.233	4.092	30.722
SAD	5.987	2.995	7.637	19.961	1.847	0.924	7.376	8.300
SWJ	46.220	23.111	5.293	15.390	14.437	7.219	5.097	13.740
UW	15.643	7.823	5.042	11.859	4.877	2.439	3.998	6.386

performance of PCapN and DTCM. Finally, it concludes with an elucidation of the case study.

A. Experimental Setup

This section gives the dataset and implementation details.

1) Dataset Description: Following [17] [18] [23] [24] [25] [34] [45], we use the University of East Anglia MTS archive (UEA2018) for performance evaluation. UEA2018 is one of the most authoritative MTS archives [81], including 30 datasets from seven application areas, i.e., audio spectra, electrocardiogram, electroencephalogram, human activity recognition, motion, eagnetoencephalography and other. Table I shows the datasets' details.

2) Implementation Details: Firstly, we introduce the parameter settings of PCapN. In the first block of Head_A, the CNN's kernel size, stride size, and channel number are set to 9, 2, and 128, respectively. In the second block of Head_A, they are set to 9, 1, and 128, respectively. The CNN's kernel size, stride size, and channel number are set to 7, 2, and 128, respectively in the first block of Head_B, while they are set to 7, 1, and 128, respectively in the second block of Head_B. In the GRN, each capsule-based transformer block has 8 capsule-based attention modules, i.e., $n_{att} = 8$, and each fully-connected layer owns a unit size of 96.

Secondly, we describe the parameter settings of LCapN. In the first block of LFN, the CNN's kernel size, stride size, and channel number are set to 9, 2, and 64, respectively. In the second block of LFN, they are set to 9, 1, and 64, respectively. In the GRN, we set $n_{att} = 8$ and each fully-connected layer owns a unit size of 24. Table II collects the results of FLOPs, parameter count, and inference time with the PCapN and LCapN on various testing datasets. LCapN is much smaller than PCapN, in terms of parameter count and FLOPs. For instance, on the ArticularyWordRecognition (AWR) dataset, the number of parameters and FLOPs for PCapN are 11.143M and 22.283M, whereas those for LCapN are 3.479M and 6.957M. Besides, on most datasets, the inference time of PCapN is significantly longer than that of LCapN, e.g., when utilizing CPU on the CharacterTrajectories (CT) dataset, the inference time results of the PCapN and LCapN are 49.717s and 18.141s, respectively. These results convincingly demonstrate that PCapN requires more computational resources than LCapN.

Last but not least, we use the Adam Optimizer to optimize the PCapN's and LCapN's parameters. The weight decay and initial learning rate values are set to 0.0005 and 0.001, respectively. We use the L_2 regularization to avoid overfitting during the training process. We run all experiments on a computer with Ubuntu 18.04 OS, one Nvidia GTX 1080Ti GPU with 11GB, and one AMD R5 1400 CPU with 16G RAM. The whole experiments took over 45 days.

B. Performance Metrics

To evaluate the performance of various MTSC algorithms, we adopt the 'win'/'tie'/'lose', mean accuracy (MeanACC), and AVG_rank as performance metrics. These metrics are statistics based on the top-1 accuracy results obtained. For an arbitrary algorithm, its 'win', 'tie', and 'lose' values reflect on how many datasets it is better than, equivalent to, and worse than the other algorithms, respectively; its 'best' score is the summation of the corresponding 'win' and 'tie' scores. Besides, we use AVG_rank to rank all compared algorithms, where the results are based on the Wilcoxon signed-rank test with Holm's alpha (5%) correction [34] [35] [37] [44].

TABLE III
TOP-1 ACCURACY RESULTS OF DIFFERENT ALGORITHMS ON 7 UEA DATASETS.

Dataset Index	LFN								LRN	GRN w/o Routing	Vanilla Trans	Embed Trans	TimeBERT	TimeGPT	GRN	PCapN
	Head_A-(1)	Head_A	Head_A-(3)	Head_A w/o Routing	Head_B-(1)	Head_B	Head_B-(3)	Head_B w/o Routing								
BM	0.8	0.975	0.975	0.675	0.6	0.85	0.85	0.65	0.975	0.9	0.9	0.95	**1**	**1**	0.95	**1**
DDG	0.3	0.4	0.4	0.32	0.4	0.5	0.6	0.3	0.54	0.56	0.5	0.56	0.56	0.54	0.58	**0.74**
FM	0.58	0.6	0.61	0.45	0.58	0.6	0.6	0.4	0.63	0.58	0.58	0.58	0.6	0.54	0.6	**0.66**
LIB	0.267	0.311	0.317	0.267	0.25	0.283	0.289	0.267	0.361	0.306	0.567	0.667	0.667	0.633	0.706	**0.789**
NATO	0.639	0.706	0.717	0.517	0.6	0.667	0.689	0.467	0.889	0.567	0.889	0.889	0.733	0.889	0.9	**0.933**
PEMS	0.359	0.651	0.669	0.431	0.338	0.376	0.401	0.327	0.827	0.699	0.827	0.847	0.847	0.827	0.908	**0.931**
UW	0.716	0.791	0.8	0.684	0.7	0.716	0.775	0.596	0.816	0.469	0.684	0.816	0.775	0.684	0.841	**0.856**
MeanACC	0.523	0.633	0.641	0.478	0.495	0.570	0.601	0.430	0.720	0.583	0.707	0.758	0.740	0.730	0.784	**0.844**

C. Effectiveness of PCapN

This section first provides an ablation study on the PCapN and then compares it with a number of MTSC algorithms.

1) Ablation Study: As aforementioned, The LFN and GRN in the PCapN are used for local-feature and global-relation extraction, respectively. To study the impact of LFN and GRN on the performance of PCapN, we evaluate their performance on 7 popular datasets, i.e., BasicMotions (BM), DuckDuckGeese (DDG), FingerMovements (FM), Libras (LIB), NATOPS (NATO), PEMSF (PEMS), and UWaveGuestureLibrary (UM).

Local Feature Network The LFN includes Head_A and Head_B, each of which stacks two squash CNN blocks and one dynamic routing block together. To study the effectiveness of these heads, we compare eight LFN variants listed below:

- $Head\_A$-(1): containing one squash CNN block and one dynamic routing block, where each block's CNN is with a kernel size of 9.
- $Head\_A$: containing two squash CNN blocks and one dynamic routing block, where each block's CNN is with a kernel size of 9.
- $Head\_A$-(3): containing three squash CNN blocks and one dynamic routing block, where each block's CNN is with a kernel size of 9.
- $Head\_A$ w/o Routing: $Head\_A$ without the dynamic routing block.
- $Head\_B$-(1): containing one squash CNN blocks and one dynamic routing block, where each block's CNN is with a kernel size of 7.
- $Head\_B$: containing two squash CNN blocks and one dynamic routing block, where each block's CNN is with a kernel size of 7.
- $Head\_B$-(3): containing three squash CNN blocks and one dynamic routing block, where each block's CNN is with a kernel size of 7.
- $Head\_B$ w/o Routing: $Head\_B$ without the dynamic routing block.
- LFN: the combination of $Head\_A$ and $Head\_B$.

The top-1 accuracy results obtained by various algorithms on 7 UEA2018 datasets are shown in Table III. First, as the number of squash CNN blocks increases, the accuracy of $Head\_A$ becomes higher and higher. This is because $Head\_A$ with multiple squash CNN blocks can capture richer local representations from the data. Meanwhile, the MeanACC value of $Head\_A$-(3) is only 0.08 higher than $Head\_A$, reflecting that $Head\_A$ and $Head\_A$-(3) have similar performance in some way. Compared with $Head\_A$, $Head\_A$-(3) needs to consume more computational resources. This is why we choose $Head\_A$ to own two squash CNN blocks rather than three. The same experimental phenomena also appear in $Head\_B$. This is why Head_B has two squash CNN blocks instead of three.

Second, it is easily observed that the $Head\_A$ and $Head\_B$ outperform the $Head\_A$ w/o Routing and $Head\_B$ w/o Routing, respectively. This is because the dynamic routing mechanism promotes the interaction among capsules to further extract the potential relationships among them. Compared with individual $Head\_A$ or $Head\_B$, it is more effective for the LFN to discover the multi-scaled local features from data. That is why the LFN is used as the local-feature extractor for the PCapN.

Global Relation Network In the PCapN, GRN includes two capsule-based transformer blocks and one dynamic routing block. To investigate the effectiveness of GRN, we compare six GRN variants listed below:

- GRN: containing two capsule-based transformer blocks and one dynamic routing block.
- GRN w/o Routing: GRN without the dynamic routing block.
- VanillaTrans: containing two vanilla transformer blocks [57] [58] and one dynamic routing block.
- EmbedTrans: containing two embedding transformer blocks [59] and one dynamic routing block.
- TimeBERT: a modified time series bidirectional transformer [82] model containing three transformer blocks.
- TimeGPT: a modified time series generative pre-training transformer [83] model containing three transformer blocks.

Clearly, GRN performs better than the GRN w/o Routing on all seven datasets. This proves again that the dynamic routing block is crucial for both the LFN and GRN, especially when extracting the relationships among capsules. The GRN obtains better MeanACC results than the VanillaTrans, EmbedTrans, TimeBERT, and TimeGPT. This is because the capsule-based transformer is able to capture sufficient intrinsic connections and regulations hiding in data by relating the capsules at different locations of the given MTS. Thanks to LFN and GRN, PCapN achieves the best performance model.

2) Comparisons and Analysis: Table IV shows the top-1 accuracy results obtained by different MTSC algorithms on all

TABLE IV
TOP-1 ACCURACY RESULTS OF VARIOUS MTSC ALGORITHMS ON 30 DATASETS.

Dataset Index	Existing SOTA	Best:DTW [19]	Best:DTWN [19]	Best:EDN [15]	XEM [19]	XM [34]	RM [15]	WM [32]	CBOSS [24]	MLCapN [44]	RISE [18]	TSF [25]	TapNet [45]	MUSE [32]	FM [15]	Multi-head Capsule [54]	Supervised RTFN [34]	Ours	
																		LCapN	PCapN
AWR	0.99	0.987	0.987	0.97	**0.993**	0.99	0.99	**0.993**	0.99	0.957	0.963	0.953	0.987	**0.993**	0.986	0.953	**0.993**	0.71	0.96
AF	0.267	0.267	0.267	0.267	0.467	0.4	0.333	0.267	0.267	0.333	0.267	0.2	0.333	0.4	0.2	**0.533**	**0.533**	0.4	**0.533**
BM	**1**	**1**	**1**	0.675	**1**	**1**	**1**	**1**	**1**	0.875	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**	**1**
CT	0.986	**1**	0.969	0.964	0.979	0.983	0.985	0.99	0.986	0.917	0.986	0.931	0.997	0.986	0.993	0.49	0.993	0.335	0.42
CK	–	–	**1**	0.944	0.986	0.972	0.986	0.986	–	–	–	–	0.958	–	0.986	**1**	0.986	0.958	0.972
DDG	0.48	0.58	0.6	0.275	0.375	0.4	0.4	0.575	0.48	0.38	0.22	0.46	0.575	0.58	0.675	0.58	0.6	0.6	**0.74**
EW	0.749	0.517	0.618	0.55	0.527	0.55	**1**	0.89	0.511	0.33	0.626	0.712	0.489	–	0.809	–	0.685	–	–
EP	**1**	**1**	0.978	0.667	0.986	0.978	0.986	0.993	0.979	0.732	0.979	**1**	0.971	0.993	0.964	0.732	0.978	0.877	0.92
EC	**0.882**	0.361	0.323	0.293	0.372	0.422	0.433	0.316	0.304	0.373	0.445	0.487	0.323	0.476	0.274	0.323	0.38	0.293	0.32
ER	0.97	0.926	0.133	0.133	0.2	0.133	0.133	0.133	0.919	0.941	0.881	0.859	0.133	**0.974**	0.133	0.919	0.941	0.922	0.952
FD	0.656	0.529	0.529	0.519	0.614	0.629	0.614	0.545	0.513	0.555	0.64	0.508	0.556	0.631	0.555	0.614	0.67	0.669	**0.679**
FM	0.582	0.53	0.53	0.55	0.59	0.53	0.569	0.54	0.519	0.58	0.581	0.562	0.53	0.551	0.61	0.58	0.63	0.62	**0.66**
HMD	0.414	0.224	0.306	0.279	0.649	0.541	0.5	0.378	0.292	0.544	0.481	0.312	0.378	0.362	0.378	0.662	0.662	0.662	**0.703**
HW	0.478	0.601	**0.607**	0.531	0.287	0.267	0.267	0.531	0.504	0.305	0.359	0.191	0.357	0.518	0.547	0.343	0.454	0.112	0.191
HB	0.64	0.604	0.717	0.62	0.761	0.693	0.8	0.727	0.564	0.458	0.535	0.518	0.751	0.515	0.714	**0.785**	**0.785**	0.64	**0.785**
IW	–	–	0.115	0.128	0.228	0.237	0.224	–	–	–	–	–	0.208	–	0.105	0.202	**0.467**	0.109	0.202
JV	–	–	0.959	0.949	0.978	0.968	0.97	0.978	–	–	–	–	0.965	–	**0.992**	0.949	0.973	0.951	0.976
LIB	0.9	0.883	0.894	0.833	0.772	0.767	0.783	0.894	0.894	0.85	0.806	0.806	0.85	0.894	**0.922**	0.767	**0.922**	0.522	0.789
LSST	0.391	0.458	0.575	0.456	**0.652**	0.633	0.612	0.628	0.458	0.39	0.161	0.265	0.568	0.435	0.646	0.367	0.451	0.257	0.333
MI	0.61	0.59	0.51	0.51	0.6	0.46	0.55	0.5	0.39	0.51	0.48	0.55	0.59	–	0.53	0.6	0.6	0.6	**0.65**
NATO	0.889	0.883	0.883	0.85	0.916	0.9	0.911	0.883	0.85	0.9	0.8	0.839	0.939	0.906	0.961	0.9	**0.967**	0.861	0.933
PD	0.941	0.977	0.977	0.939	0.977	0.951	0.951	0.969	0.939	0.979	0.892	0.831	0.98	0.967	**0.987**	0.939	**0.987**	0.911	0.944
PEMS	0.981	0.981	0.734	0.705	0.942	0.983	0.983	–	0.73	0.745	0.982	**0.994**	0.751	–	0.653	0.925	0.936	0.913	0.931
PS	0.321	0.195	0.151	0.104	0.288	0.187	0.222	0.19	0.151	0.151	0.137	0.269	0.175	–	0.275	**0.33**	**0.33**	0.127	**0.33**
RS	0.898	0.891	0.868	0.842	**0.941**	0.928	0.921	0.914	0.854	0.856	0.895	0.823	0.868	0.933	0.882	0.842	0.862	0.875	0.882
SRS1	0.854	0.806	0.775	0.771	0.839	0.829	0.826	0.744	0.765	0.908	0.84	0.724	0.652	0.697	0.867	0.867	**0.922**	0.899	0.913
SRS2	0.533	0.539	0.539	0.533	0.55	0.483	0.478	0.522	0.533	0.506	0.483	0.494	0.55	0.528	0.522	0.611	**0.622**	0.583	0.589
SAD	–	–	0.963	0.967	0.973	0.97	0.968	0.982	–	–	–	–	0.983	–	**0.994**	0.957	0.986	0.787	0.946
SWJ	0.467	0.333	0.333	0.2	0.4	0.333	0.467	0.333	0.333	0.4	0.333	0.267	0.4	0.267	0.467	0.6	**0.667**	0.533	**0.667**
UW	0.897	0.903	0.903	0.881	0.897	0.894	0.9	0.903	0.869	0.859	0.775	0.684	0.894	**0.931**	0.857	0.838	0.903	0.806	0.856
Best	3	3	3	0	4	1	2	2	1	0	1	3	3	4	5	5	**12**	1	10
AVG_Rank	8.283	10.117	10.133	13.867	7.100	9.767	8.183	9.383	13.100	12.467	12.400	13.617	9.567	10.600	8.167	9.667	4.400	11.283	7.900

the 30 UEA2018 datasets. For each dataset, the existing SOTA stands for the best-performance algorithm on that dataset, covering HULM [22], gRSF [31], STC [15], and so on; similarly, Best:DTW, Best:DTWN, and Best:EDN are the best DTW-based (e.g., DTW_I and DTW_A in [19]), DTWN-based (e.g., $DTW\text{-}1NN_I(n)$ and $DTW\text{-}1NN_D(n)$ [19].) and ED-NN-based (e.g., ED-1NN and ED-1NN (Normalized) [15]) algorithms on that dataset, respectively. Note that the PCapN, LCapN, and multi-head capsule [54] are not validated on the EW dataset due to the limited computing resources currently available to the authors.

It is seen that the supervised RTFN and PCapN take the first and second positions in the competition since their 'best' scores are 12 and 10. Both the FM (i.e., FCN-Multivariate LSTM) and multi-head capsule rank the third as they have a 'best' score of 5. Meanwhile, the supervised RTFN, XEM, and PCapN are the best, second-best, and third-best algorithms in terms of AVG_rank. Best:EDN is the worst algorithm with the 'best' and AVG_rank metrics jointly considered.

The following explains the reasons behind the findings above. The dual-network-based structures, including the supervised RTFN, FM, and PCapN, take advantage of two parallel networks, one for local-feature extraction and the other for global-relation extraction. Inspired by the idea of 'divide-and-conquer', these structures can well handle different feature-extraction tasks simultaneously. This, to a certain extent, supports our motivation to focus on dual-network-based student and teacher models.

The XEM, as a state-of-the-art MTSC algorithm, uses an explicit boosting-bagging approach to capture the latent relationships among dimensions at different timestamps. The multi-head capsule algorithm relies on multiple capsule network to capture the multi-scaled features from MST data, where capsules are used to compensate for the loss caused by the translation invariance of neural networks, e.g., Maxpooling. On the contrary, it is quite challenging for the Best:EDN to simultaneously emphasize the useful representations from MTS data and the relationships among them as DTW-1NN-based approaches are not good at discovering the internal representational hierarchy of the MTS data.

D. Effectiveness of DTCM

As introduced in Section III, DTCM is a targeted and offline distillation method based on mutual learning for dual-network-based student and teacher models. This section first offers an ablation study on the DTCM and then compares it with 19 popular KD algorithms. Following references [58] [60] [61] [62] [63] [64] [68] [69] [70] [71] [72] [73] [74] [75] [76] [77] [78] [79] [80], we use the LCapN's performance to indicate that of a KD algorithm deployed for knowledge transfer between the PCapN and LCapN.

1) Ablation Study: The 7 datasets in Table III are used to study the impact of important parameter settings on the DTCM's performance.

DTCM with different Γ ***values*** Γ is a temperature coefficient for producing a soft probability distribution over classes. Table V shows the top-1 accuracy results of DTCM with different Γ values. It is easily seen that $\Gamma = 1$ is the best setting as it helps the DTCM to obtain the highest top-1 accuracy value on each dataset.

Offline or Online Table VI shows the top-1 accuracy results of offline/online DTCM. Compared with the offline

TABLE V
TOP-1 ACCURACY RESULTS OF DTCM WITH DIFFERENT Γ VALUES ON 7 UEA DATASETS.

Dataset Index	DTCM						
	Γ = 0.5	Γ = 1	Γ = 2	Γ = 3	Γ = 4	Γ = 5	Γ = 10
BM	0.975	**1**	**1**	**1**	**1**	**1**	**1**
DDG	0.68	**0.72**	0.7	0.7	0.7	0.7	0.7
FM	0.6	**0.64**	0.62	0.62	0.62	0.6	0.6
LIB	0.722	**0.772**	**0.772**	0.744	0.744	0.733	0.733
NATO	0.9	**0.95**	**0.95**	0.9	0.9	0.9	0.9
PEMS	**0.948**	**0.948**	**0.948**	**0.948**	**0.948**	0.931	0.913
UW	0.889	**0.9**	**0.9**	**0.9**	**0.9**	**0.9**	**0.9**
MeanACC	0.814	**0.847**	0.815	0.802	0.802	0.794	0.791

TABLE VI
TOP-1 ACCURACY RESULTS OF OFFLINE/ONLINE DTCM ON 7 UEA DATASETS.

Dataset Index	DTCM	
	Online	Offline
BM	**1**	**1**
DDG	**0.72**	**0.72**
FM	**0.68**	0.64
LIB	**0.772**	**0.772**
NATO	**0.95**	**0.95**
PEMS	**0.948**	**0.948**
UW	**0.9**	**0.9**
MeanACC	**0.853**	0.847

DTCM, the online DTCM takes less training time on each dataset, e.g., it costs 32 mins for the online DTCM to train its PCapN and LCapN models on the BM dataset while the training time increases to 48.3 mins for the offline DTCM. This is because, the online DTCM involves pre-training, providing the PCapN with sufficient prior knowledge that effectively shortens the training time. However, pre-training is usually time-consuming and requires much computing resources. Meanwhile, the performance of offline and online methods is almost the same on the 7 datasets. The online DTCM is slightly better than the offline DTCM on dataset FM. Hence, this paper adopts the offline DTCM.

DTCM with different KD Losses For an arbitrary KD algorithm, it is crucial to select an appropriate KD loss to measure the average difference between the outputs of student and teacher models. To study the impact of KD loss on the DTCM's performance, we collect the top-1 accuracy results of DTCM with different KD losses in Table VII. Obviously, L_2 performs better than the others. Hence we use the L_2 loss to promote the knowledge transfer between the PCapN and LCapN.

2) Comparisons and Analysis: To thoroughly evaluate the performance of DTCM, we compare it with 19 state-of-the-art KD algorithms on 29 UEA datasets against the 'best', MeanACC, and AVG_rank values. Note that all algorithms are not validated on dataset EW due to the limited computing resources the authors can access to. The top-1 accuracy results are shown in Table VIII. It is observed that the DTCM performs the best among all KD algorithms for comparison since it obtains the highest MeanACC and 'best' values, namely, 0.747 and 13, and the smallest AVG_rank score,

TABLE VII
TOP-1 ACCURACY RESULTS OF DTCM WITH DIFFERENT KD LOSSES ON 7 UEA DATASETS. ABBREVIATIONS: L_1–L_1 LOSS, HL–HUGE LOSS, CE–CROSS ENTROPY, KL–KULLBACK LEIBLER, FN–FROBENIUS NORM, EMD– EARTH MOVER DISTANCE, L_2–L_2 LOSS (I.E., THE MEAN SQUARE ERROR).

Dataset Index	DTCM						
	L_1	HL	CE	KL	FN	EMD	L_2
BM	**1**	0.95	0.925	0.975	0.975	**1**	**1**
DDG	0.48	0.6	0.54	0.62	0.6	0.56	**0.72**
FM	0.55	0.6	0.59	0.59	0.6	0.58	**0.64**
LIB	**0.8**	0.717	0.744	0.739	0.743	0.717	0.772
NATO	0.9	0.85	0.85	0.872	0.883	0.867	**0.95**
PEMS	0.896	0.892	0.873	0.888	0.878	0.881	**0.948**
UW	0.878	0.869	0.866	0.869	0.869	0.857	**0.9**
MeanACC	0.786	0.783	0.770	0.793	0.793	0.780	**0.847**

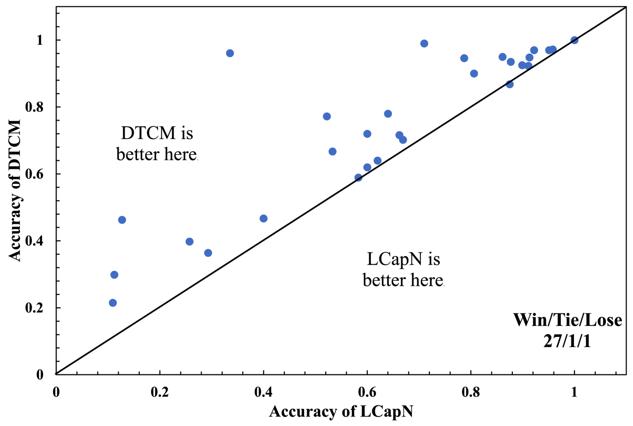


Fig. 4. Accuracy plot showing the performance difference between DTCM and LCapN on 29 UEA datasets.

namely 3.879. In the meantime, the DFA ranks the second according to the MeanACC value, namely, 0.729, while the LSN takes the second place against AVG_rank , namely, 6.914. On the other hand, KR achieves the worst performance among all in terms of MeanACC and AVG_rank, namely, 0.511, and 16.534.

The following explains the reasons behind our observations above. The DTCM is a targeted distillation method for the dual-network-based PCapN and LCapN, where they are both composed of two parallel networks with different tasks, one as a local-feature extractor and the other as a global-relation extractor. Based on mutual learning, this method enables efficient knowledge transfer between the teacher and student via two- and one-way KD operations, which helps the LCapN to capture sufficient intrinsic connections and regulations hidden in the data. The DFA uses a differentiable feature aggregation search method to discover the proper connections between the PCapN and LCapN, while the LSN, based on feature embedding, encourages the LCapN to inherit the high-level features extracted by the PCapN. The KR algorithm is suitable for the distillation between single-network-based teacher and student models. However, The LCapN with KR leads to poor performance since the KR's knowledge aggregation scheme

TABLE VIII
TOP-1 ACCURACY RESULTS OF VARIOUS KD ALGORITHMS ON 29 UEA DATASETS.

Dataset Index	LCapN	Response-based KD			Feature-based KD						Relation-based KD						Other KD				DTCM
	(Baseline)	DKNN [60]	FSP [84]	CTL [70]	FitNet [72]	KR [73]	FT [74]	RL [75]	NST [85]	DFA [76]	LMTN [86]	SP [77]	MLKD [78]	IRG [79]	LSN [80]	RKT [87]	FEKD [88]	Tf-KD [62]	DML [63]	DCM [64]	
AWR	0.71	0.5	0.457	0.817	0.587	0.647	0.973	0.976	0.523	0.967	0.867	0.667	0.967	0.973	0.973	0.976	0.917	0.967	0.95	0.973	**0.99**
AF	0.4	0.467	**0.533**	0.467	0.467	0.467	**0.533**	0.4	**0.533**	**0.533**	0.4	**0.533**	0.4	0.467	0.4	0.333	0.4	0.467	0.467	0.467	0.467
BM	**1**	0.725	0.85	0.925	0.8	0.625	0.875	**1**	0.6	**1**	0.8	0.85	0.65	0.925	**1**	0.975	**1**	**1**	0.95	**1**	**1**
CT	0.335	0.341	0.121	0.941	0.264	0.115	0.936	0.934	0.129	0.933	0.218	0.348	0.577	0.958	0.937	0.937	**0.971**	**0.971**	0.899	0.906	0.961
CK	0.958	0.375	0.347	**0.986**	0.236	0.389	**0.986**	**0.986**	0.257	0.944	0.5	0.556	0.642	0.944	0.958	0.958	0.972	0.986	0.931	0.861	0.972
DDG	0.6	0.36	0.4	0.64	0.4	0.36	0.62	0.66	0.3	0.64	0.38	0.32	0.32	0.62	0.66	0.62	0.6	0.56	0.46	0.6	**0.72**
EP	0.877	0.478	0.609	0.906	0.468	0.493	0.906	0.891	0.565	0.87	0.522	0.442	0.529	0.87	0.933	0.877	0.862	0.804	0.725	0.758	**0.935**
EC	0.293	0.279	0.282	0.297	0.315	0.27	0.254	0.327	0.314	0.336	0.319	0.332	0.315	0.318	0.338	**0.367**	0.335	0.255	0.319	0.323	0.364
ER	0.922	0.789	0.715	0.937	0.87	0.933	0.922	0.952	0.852	0.956	0.848	0.885	0.87	0.952	0.952	0.948	0.967	0.967	**0.97**	**0.97**	**0.97**
FD	0.669	0.53	0.545	0.7	0.684	0.537	0.666	0.697	0.518	0.694	0.575	0.548	0.527	0.695	0.702	0.685	0.685	0.685	0.684	0.69	**0.702**
FM	0.62	0.58	0.63	0.61	0.59	0.54	0.57	0.59	0.53	0.58	0.58	0.59	0.63	0.6	0.56	0.67	0.61	0.64	0.57	0.59	0.64
HMD	0.662	0.5	0.554	0.689	0.527	0.431	0.621	**0.73**	0.419	0.732	0.5	0.446	0.441	**0.73**	**0.73**	0.649	0.703	0.716	0.607	0.69	0.716
HW	0.112	0.107	0.216	0.216	0.213	0.1	0.228	0.227	0.222	0.24	0.107	0.115	0.218	0.209	0.233	0.268	0.256	**0.362**	0.211	0.216	0.299
HB	0.64	0.722	0.727	**0.78**	0.755	0.731	0.761	0.741	0.735	0.721	0.741	0.721	0.721	0.74	0.731	**0.78**	0.731	0.74	0.776	**0.78**	**0.78**
IW	0.109	0.106	0.103	0.163	0.104	0.108	0.285	0.329	0.105	0.386	0.108	0.104	0.107	0.377	0.34	0.393	**0.436**	0.393	0.162	0.184	0.215
JV	0.951	0.8	0.749	0.954	0.67	0.622	0.96	**0.983**	0.805	0.968	0.598	0.751	0.624	0.962	0.954	0.912	0.953	0.954	0.935	0.932	0.97
LIB	0.522	0.433	0.494	0.756	0.5	0.478	0.772	**0.806**	0.467	0.772	0.411	0.4	0.444	0.811	0.789	0.8	0.767	0.767	0.683	0.756	0.772
LSST	0.257	0.336	0.332	0.349	0.346	0.348	0.378	0.372	0.36	0.358	0.345	0.346	0.347	0.364	0.399	0.396	**0.402**	0.384	0.361	0.368	0.398
MI	0.6	**0.65**	0.58	0.62	0.41	0.52	0.62	0.58	0.5	0.5	0.5	0.61	0.53	0.59	0.43	0.62	0.55	0.59	0.54	0.58	0.62
NATO	0.861	0.828	0.667	0.872	0.872	0.661	0.889	0.894	0.717	0.917	0.606	0.683	0.672	0.906	0.906	0.906	0.928	0.9	0.878	0.928	**0.95**
PD	0.911	0.61	0.842	0.647	0.67	0.697	0.932	0.935	0.571	0.948	0.864	0.593	0.84	0.947	0.949	0.958	0.963	**0.965**	0.896	0.935	0.923
PEMS	0.913	0.908	0.867	0.936	0.843	0.741	0.896	0.948	0.803	**0.954**	0.717	0.838	0.885	0.942	0.937	0.903	0.919	0.908	0.874	0.881	0.948
PS	0.127	0.462	0.462	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	**0.463**	0.444	0.462	**0.463**
RS	0.875	0.612	0.559	**0.895**	0.684	0.691	0.847	0.888	0.572	0.875	0.572	0.651	0.678	0.849	**0.895**	0.855	0.827	0.868	0.862	0.855	0.868
SRS1	0.899	0.9	0.895	0.888	0.898	0.816	0.815	0.894	0.917	0.894	0.887	0.899	0.897	0.897	0.91	0.921	0.903	0.917	0.893	0.904	**0.925**
SRS2	0.583	0.507	0.583	0.556	**0.589**	0.556	0.528	0.506	0.55	0.583	0.527	0.583	0.572	0.544	0.572	0.544	0.573	0.55	0.544	0.544	**0.589**
SAD	0.787	0.204	0.211	0.896	0.237	0.268	0.957	0.96	0.22	0.953	0.953	0.946	0.946	0.964	**0.97**	0.959	**0.97**	0.967	0.858	0.953	0.946
SWJ	0.533	0.467	0.216	0.533	0.533	0.533	0.6	0.4	0.6	0.533	0.533	0.6	0.4	0.4	0.467	0.533	0.467	0.333	0.533	0.6	**0.667**
UW	0.806	0.694	0.706	0.806	0.6	0.675	0.888	0.878	0.731	0.897	0.647	0.687	0.778	0.891	0.888	0.866	0.891	0.884	0.822	0.859	**0.9**
Best	1	1	1	4	2	1	3	6	2	4	1	1	1	2	5	3	6	5	1	3	**13**
MeanACC	0.639	0.527	0.526	0.698	0.538	0.511	0.713	0.722	0.513	0.729	0.555	0.569	0.586	0.721	0.723	0.727	0.725	0.723	0.683	0.709	**0.747**
AVG_Rank	12.052	16.345	15.689	9.207	14.655	16.534	9.259	7.569	15.5	7.328	16.138	14.224	14.759	8.138	6.914	7.034	7.293	7.397	11.983	9.103	**3.879**

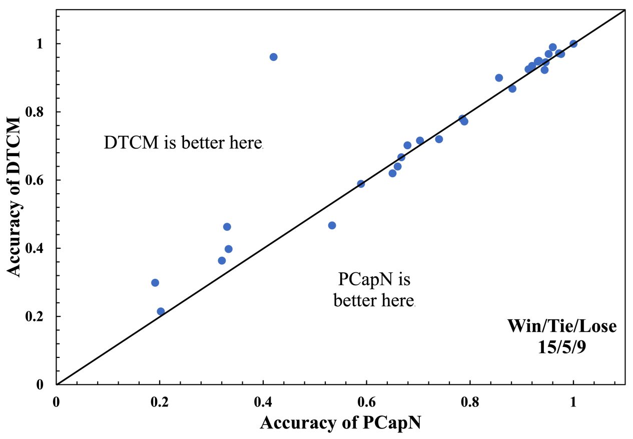


Fig. 5. Accuracy plot showing the performance difference between DTCM and PCapN on 29 UEA datasets.

is not that effective to model the parameter distributions for dual-network-based teacher and student models.

To visualize the differences between DTCM and LCapN, Fig. 4 depicts the accuracy plot of DTCM against DCM on 29 UEA datasets. The result shows that DTCM achieves 27/1/1 in terms of 'win'/'lose'/'tie' measure, which reflects that the proposed DTCM provides adequate knowledge transfer between PCapN and LCapN, effectively improving the performance of LCapN. Even compared with PCapN, DTCM has some advantages. We show the accuracy plot of DTCM against PCapN on 29 UEA datasets in Fig. 5. DTCM obtains 'win'/'lose'/'tie' in 15/5/9 cases, respectively, reflecting the effectiveness of DTCM for dual-network-based student and teacher knowledge transfer.

E. Case Study

To further investigate the impact of DTCM deployed for knowledge transfer between the PCapN and LCapN, we use the unsupervised t-distributed stochastic neighbor embedding (t-SNE) [89] approach to visualize the representations learned by PCapN, LCapN, and DTCM. Fig. 6 displays the visualization of representations learned by t-SNE, LCapN with t-SNE, PCapN with t-SNE on the NATOPS (NATO) and UWaveGestureLibrary (UW) datasets.

In Fig. 6, it is apparent that LCapN and PCapN are more adept at consolidating data with similar characteristics than t-SNE, showcasing the remarkable feature extraction capabilities inherent in both models. Meanwhile, PCapN achieves a better clustering effect than LCapN, reflecting a stronger feature extraction prowess. To enhance the feature extraction capabilities of LCapN, DTCM is used to transfer the knowledge from PCapN to LCapN. Upon closer examination of Fig. 6, one can easily find that DTCM substantially enhances the clustering effect of LCapN when compared with its standalone version. This demonstrates that DTCM effectively facilitates knowledge transfer between the PCapN and LCapN, helping LCapN mine rich regularizations and relationships within the data.

V. CONCLUSION

In the proposed PCapN, the LFN and GRN can capture sufficient local and global patterns of MTS data and discover the intrinsic relationships among these patterns. The proposed

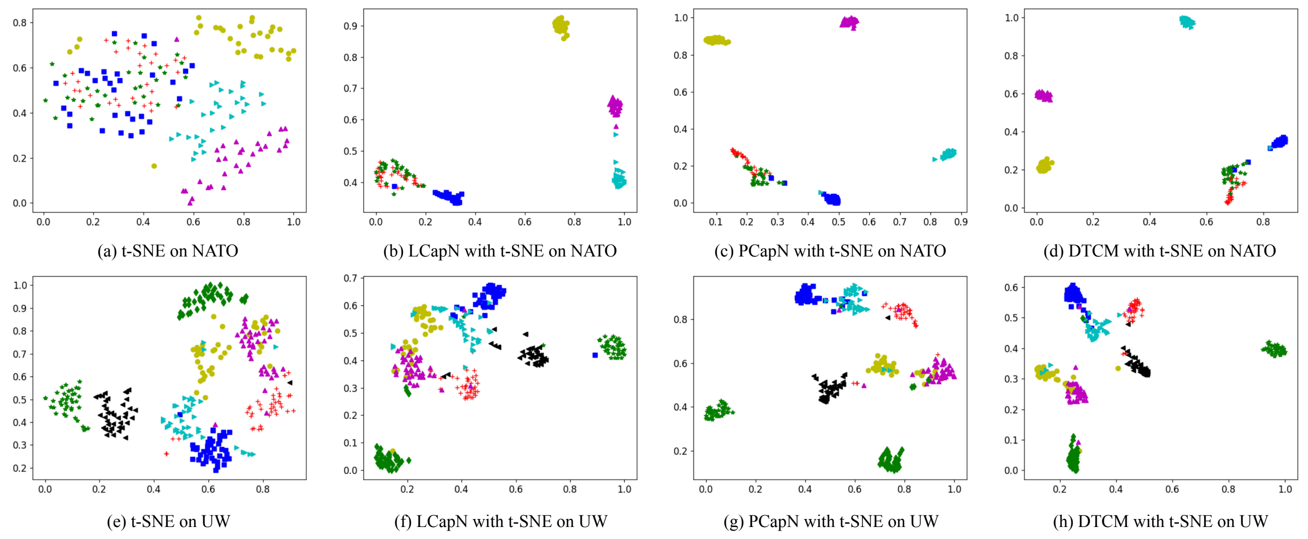


Fig. 6. Visualization of representations learned by t-SNE, LCapN with t-SNE, PCapN with t-SNE on the NATOPS (NATO) and UWaveGestureLibrary (UW) datasets.

DTCM offers mutual-learning-based targeted and offline distillation to supervise efficient knowledge transfer between the PCapN and LCapN. Experimental results demonstrate that compared with a large number of MTSC algorithms, the PCapN performs the best on 10 out of 30 datasets in the UEA2018 archive, regarding the 'win'/'tie'/'lose' statistics and AVG rank; compared with 19 state-of-the-art KD algorithms, the DTCM wins 13 out of 29 datasets in terms of the MeanACC, 'win'/'tie'/'lose' statistics, and AVG rank. The experimental results also indicate that the PCapN has good potential to handle various real-world MTSC problems; DTCM is a promising candidate to address KD problems for dual-network-based student and teacher models in a variety of time-series applications.

The proposed PCapN only uses a simple concatenation to fuse LFN and GRN, easily ignoring the loss of instance-level hierarchical relationships between LFN and GRN. To address this problem, we plan to propose a dynamic hierarchical feature fusion method for dual-network-based feature fusion in the next phase work. This method can explore the valid hierarchical instance-level connections between the local feature network and the global relational network in a dual-network-based model, helping the model mine the rich regularizations and relationships hidden in the data.

Knowledgement

The authors are grateful to the editors and reviews for providing us with valuable suggestions, improving the quality and completeness of the article.

References

[1] P.-Y. Zhou and K. C. C. Chan, "Fuzzy feature extraction for multichannel eeg classification," *IEEE Trans. Cogn. Dev. Syst.*, vol. 10, no. 2, pp. 267–279, 2018.
[2] Y. Zhang, H. Cai, J. Wu, L. Xie, M. Xu, D. Ming, Y. Yan, and E. Yin, "Emg-based cross-subject silent speech recognition using conditional domain adversarial network," *IEEE Trans. Cogn. Dev. Syst.*, pp. 1–9, 2023.
[3] A. M. Roy and J. Bhaduri, "Densesph-yolov5: An automated damage detection model based on densenet and swin-transformer prediction head-enabled yolov5 with attention mechanism," *Adv. Eng. Informatics*, vol. 56, pp. 1–16, 2023.
[4] F. Lin, Z. Wang, H. Zhao, S. Qiu, R. Liu, X. Shi, C. Wang, and W. Yin, "Hand movement recognition and salient tremor feature extraction with wearable devices in parkinson's patients," *IEEE Trans. Cogn. Dev. Syst.*, pp. 1–12, 2023.
[5] N. Li, F. Chang, and C. Liu, "A self-trained spatial graph convolutional network for unsupervised human-related anomalous event detection in complex scenes," *IEEE Trans. Cogn. Dev. Syst.*, vol. 15, no. 2, pp. 737–750, 2023.
[6] Y. Yan, X. Wu, C. Li, Y. He, Z. Zhang, H. Li, A. Li, and L. Wang, "Topological eeg nonlinear dynamics analysis for emotion recognition," *IEEE Trans. Cogn. Dev. Syst.*, vol. 15, no. 2, pp. 625–638, 2023.
[7] S. Jamil and A. M. Roy, "An efficient and robust phonocardiography (pcg)-based valvular heart diseases (vhd) detection framework using vision transformer (vit)," *Comp. Bio. Med.*, vol. 158, pp. 1–15, 2023.
[8] H. Wu, L. Kong, Y. Zeng, and H. Bao, "Resting-state brain connectivity via multivariate emd in mild cognitive impairment," *IEEE Trans. Cogn. Dev. Syst.*, vol. 14, no. 2, pp. 552–564, 2022.
[9] W. Wang, S. Zhang, Z. Wang, X. Luo, P. Luan, A. Hramov, J. Kurths, C. He, and J. Li, "Diagnosis of early mild cognitive impairment based on associated high-order functional connection network generated by multi-modal mri," *IEEE Trans. Cogn. Dev. Syst.*, pp. 1–11, 2023.
[10] H. Xing, Z. X. R. Qu, Z. Zhu, and B. Zhao, "An efficient federated distillation learning system for multi-task time series classification," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022.
[11] J.-Y. Franceschi, A. Dieuleveut, and M. Jaggi, "Unsupervised scalable representation learning for multivariate time series," in *Proc. Adv. neural inf. proces. syst.*, 2019, p. 4650–4661.
[12] Z. Xiao, H. Xing, R. Qu, L. Feng, S. Luo, P. Dai, B. Zhao, and Y. Dai, "Densely knowledge-aware network for multivariate time series classification," *IEEE Trans. Syst. Man Cy-S.*, pp. 1–13, 2024.
[13] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P. A. Muller, "Deep learning for time series classification: a reviewer," *Data Min. Knowl. Disc.*, vol. 33, pp. 917–963, 2019.
[14] A. Bagnall., J. Lines, A. Bostrom, J. Large, and E. Keogh, "The great time series classification bake off: a review and experimental evaluation of recent algorithmic advances," *Data Min. Knowl. Disc.*, vol. 31, pp. 1–55, 2017.
[15] A. R. Puiz, M. Flynn, and A. Bagnall, "Benchmarking multivariate time series classification algorithms," *arXiv preprint arXiv:2007.13156*, 2020.
[16] J. Lines and A. Bagnall., "Time series classification with ensembles of elastic distance measures," *Data Min. Knowl. Disc.*, vol. 29, pp. 565–592, 2015.
[17] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time series classification

with cote: the collective of transformation-based ensembles," in *Proc. IEEE Int. Conf. Data Eng. (ICDE)*, 2016, pp. 1548–1549.

[18] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with hive-cote: the hierarchical of transformation-based ensembles," *ACM Trans. Knowl. Discov. D.*, vol. 21, no. 52, pp. 1–35, 2018.

[19] K. Fauvel, É. Fromont, V. Masson, P. Faverdin, and A. Termier, "Xem: An explainable-by-design ensemble method for multivariate time series classification," *Data Min. Knowl. Disc.*, vol. 36, pp. 917–957, 2022.

[20] G. R. M. G. Baydogan and E. Tuv, "A bag-of-features framework to classify time series," *IEEE Trans. Pattern Anal.*, vol. 35, no. 11, pp. 2796–2802, 2013.

[21] J. Hills, J. Lines, E. Baranauskas, J. Mapp, and A. Bagnall, "Classification of time series by shapelet transformation," *Data Min. Knowl. Disc.*, vol. 28, p. 851–881, 2014.

[22] W. Pei, H. Dibeklioğlu, D. M. J. Tax, and L. van der Maaten, "Multivariate time-series classification using the hidden-unit logistic model," *IEEE Trans. Neur. Net. Lear.*, vol. 29, no. 4, pp. 920–931, 2018.

[23] M. G. Baydogan and G. Runger, "Time series representation and similarity based on local auto patterns," *Data Min. Knowl. Disc.*, vol. 30, p. 476–509, 2016.

[24] J. Large, A. Bagnall, S. Malinowski, and R. Tavenard, "From bop to boss and beyond: time series classification with dictionary based classifier," *arXiv preprint arXiv:1809.06751*, 2018.

[25] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Inform. Sciences*, vol. 239, p. 142–153, 2013.

[26] A. Shifaz, C. Pelletier, F. Petitjean, and G. Webb, "Ts-chief: A scalable and accurate forest algorithm for time series classification," *Data Min. Knowl. Disc.*, vol. 34, p. 742–775, 2020.

[27] C. Ji, M. Du, Y. Hu, S. Liu, L. Pan, and X. Zheng, "Time series classification based on temporal features," *App. Soft Comput.*, vol. 128, pp. 1–11, 2022.

[28] K. Wu, K. Yuan, Y. Teng, J. Liu, and L. Jiao, "Broad fuzzy cognitive map systems for time series classification," *App. Soft Comput.*, vol. 128, pp. 1–13, 2022.

[29] M. G. Baydogan and G. Runger, "Learning a symbolic representation for multivariate time series classification," *Data Min. Knowl. Disc.*, vol. 29, p. 400–422, 2015.

[30] K. S. Tuncel and M. G. Baydogan, "Autoregressive forests for multivariate time series modeling," *Pattern Recogn.*, vol. 73, pp. 202–215, 2018.

[31] I. Karlsson, P. Papapetrou, and H. Boström, "Generalized random shapelet forests," *Data Min. Knowl. Disc.*, vol. 30, pp. 1053–1083, 2016.

[32] P. Schäfer and U. Leser, "Multivariate time series classification with weasel+muse," *arXiv preprint arXiv:1711.11343*, 2017.

[33] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, pp. 436–444, 2015.

[34] Z. Xiao, X. Xu, H. Xing, S. Luo, P. Dai, and D. Zhan, "Rtfn: A robust temporal feature network for time series classification," *Inform. Sciences*, vol. 571, pp. 65–86, 2021.

[35] Y. Zheng, Q. Liu, E. Chen, Y. Ge, and J. Zhao, "Time series classification using multi-channels deep convolutional neural networks," in *Proc. Lect. Notes Comput. Sci., WAIM*, 2014, pp. 298–310.

[36] K. Kashiparekh, J. Narwariya, P. Malhotra, L. Vig, and G. Shroff, "Convtimenet: A pre-trained deep convolutional neural network for time series classification," in *Proc. Int. Jt. Conf. Neural Networks (IJCNN)*, 2019, pp. 1–8.

[37] H. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. Schmidt, J. Weber, G. Webb, L. Idoumghar, P.-A. Muller, and F. Petitjean, "Inceptiontime: finding alexnet for time series classification," *Data Min. Knowl. Disc.*, vol. 34, pp. 1936–1962, 2020.

[38] A. Dempster, F. Petitjean, and G. Webb, "Rocket: exceptionally fast and accurate time series classification using random convolutional kernels," *Data Min. Knowl. Disc.*, vol. 34, p. 1454–1495, 2020.

[39] A. Dempster, D. F. Schmidt, and G. I. Webb, "Minirocket: A very fast (almost) deterministic transform for time series classification," in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2021, pp. 248–257.

[40] Z. Xiao, H. Xing, B. Zhao, R. Qu, S. Luo, P. Dai, K. Li, and Z. Zhu, "Deep contrastive representation learning with self-distillation," *IEEE Trans. Emerg. Top. Comput. Intell.*, pp. 1–13, 2023.

[41] W. Cheng, P. Sunganthan, and R. Katuwal, "Time series classification using diversified ensemble deep random vector functional link and resnet features," *Appl. Soft Comput.*, vol. 112, pp. 1–12, 2021.

[42] Z. Huang, C. Yang, X. Chen, X. Zhou, G. Chen, T. Huang, and W. Gui, "Functional deep echo state network improved by a bi-level optimization approach for multivariate time series classification," *Appl. Soft Comput.*, vol. 106, pp. 1–12, 2021.

[43] S. H. Huang, L. Xu, and C. Jiang, "Residual attention net for superior cross-domain time sequence modeling," *Fintech with Artificial Intelligence, Big Data, and Blockchain. Springer*, 2021.

[44] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.

[45] X. Zhang, Y. Gao, J. Lin, and C.-T. Lu, "Tapnet: Multivariate time series classification with attentional prototypical network," in *Proc. AAAI Conf. Artif. Intell.*, 2020, pp. 6845–6852.

[46] H. Xing, Z. Xiao, D. Zhan, S. Luo, P. Dai, and K. Li, "Selfmatch: Robust semisupervised time-series classification with self-distillation," *Int. J. Intell. Syst.*, vol. 37, no. 52, pp. 8583–8610, 2022.

[47] S. Sabour, N. Frosst, and G. E. Hinton, "Dynamic routing between capsules," in *Proc. Conf. Neural Inf. Process. Syst.*, 2017, pp. 3856–3866.

[48] K. Patrick, A. Adekoya, A. Mighty, and B. Edward, "Capsule networks – a survey," *J. King SAUD Univ. Sci.*, vol. 34, no. 1, pp. 1295–1310, 2022.

[49] Q. Zhang, B. Wu, Z. Sun, and Y. Ye, "Gating augmented capsule network for sequential recommendation," *Knowl.-Based Syst.*, vol. 247, pp. 1–13, 2022.

[50] R. Huang, J. Li, S. Wang, G. Li, and W. Li, "A robust weight-shared capsule network for intelligent machinery fault diagnosis," *IEEE Trans. Ind. Inform.*, vol. 16, no. 10, pp. 6466–6475, 2020.

[51] H. Yao, P. Gao, J. Wang, P. Zhang, C. Jiang, and Z. Han, "Capsule network assisted iot traffic classification mechanism for smart cities," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 7515–7525, 2019.

[52] H. Peng, J. Li, S. Wang, L. Wang, Q. Gong, R. Yang, B. Li., P. Yu, and L. He, "Hierarchical taxonomy-aware and attentional graph capsule rcnns for large-scale multi-label text classification," *IEEE Trans. Knowl. Data En.*, vol. 33, no. 6, pp. 2505–2519, 2021.

[53] H. Jayasekara, V. Jayasundara, J. Rajasegaran, S. Jayasekara, S. Seneviratne, and R. Rodrigo, "Timecaps: learning time series data with capsule networks," *arXiv preprint arXiv: 1911.11800*, 2019.

[54] Z. Xiao, X. Xu, H. Zhang, and E. Szczerbicki, "A new multi-process collaborative architecture for time series classification," *Knowl.-Based Syst.*, vol. 220, pp. 1–11, 2021.

[55] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Proc. Conf. Neural Inf. Process. Syst.*, 2017, p. 6000–6010.

[56] Y. Tay, M. Dehghani, D. Bahri, and D. Metzler, "Efficient transformers: a survey," *arXiv preprint arXiv: 2009.06732*, 2020.

[57] J. Liu, X. Liu, H. Lin, B. Xu, Y. Ren, Y. Diao, and L. Yang, "Transformer-based capsule network for stock movements prediction," in *Proc. FinNLP@IJCAI 2019*, 2019, pp. 66–73.

[58] T. Saha, S. R. Jayashree, S. Saha, and P. Bhattacharyya, "Bert-caps: A transformer-based capsule network for tweet act classification," *IEEE Trans. Comput. Social Sys.*, vol. 7, no. 5, pp. 1168–1179, 2020.

[59] S. Duan, J. Cao, and H. Zhao, "Capsule-transformer for neural machine translation," *arXiv preprint arXiv: 2004.14649*, 2020.

[60] G. Hinton, O. Vinyals, and J. Dean, "Distillation the knowledge in a neural network," *arXiv preprint arXiv: 1503.02531*, 2015.

[61] J. Guo, B. Yu, S. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vision*, vol. 129, p. 1789–1819, 2021.

[62] L. Yuan, F. Tay, G. Li, T. Wang, and J. Feng, "Revisit knowledge distillation via label smoothing regularization," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2020, pp. 3902–3910.

[63] Y. Zhang, T. Xiang, T. Hospedales, and H. Lu, "Deep mutual learning," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2018, pp. 4320–4328.

[64] A. Yao and D. Sun, "Knowledge transfer via dense cross-layer mutual-distillation," in *Proc. Lect. Notes Comput. Sci., ECCV*, 2020, pp. 294–311.

[65] G. Hinton, A. Krizhevsky, and S. Wang, "Transforming auto-encoders," in *Proc. Lect. Notes Comput. Sci., ICANN*, 2011, pp. 44–51.

[66] Y. Zhao, T. Birdal, H. Deng, and F. Tombari, "3d point capsule networks," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2018, pp. 1009–1018.

[67] J. Yang, P. Zhao, Y. Rong, C. Yao, C. Li, H. Ma, and J. Huang, "Hierarchical graph capsule network," in *Proc. AAAI Conf. Artif. Intell.*, 2019, pp. 1–9.

[68] G. Chen, W. Choi, X. Yu, T. Han, and M. Chandraker, "Learning efficient object detection models," in *Proc. Conf. Neural Inf. Process. Syst.*, 2017, pp. 742–751.

[69] F. Zhang, X. Zhu, and M. Ye, “Fast human pose estimation,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2019, pp. 3512–3521.

[70] Z. Meng, J. Li, Y. Zhao, and Y. Gong, “Conditional teacher-student learning,” in *Proc. IEEE Int. Conf. Acoust. Speech Signal Process. (ICASSP)*, 2019, pp. 6445–6449.

[71] R. Müller, S. Kornblith, and G. Hinton, “When does label smoothing help?” in *Proc. Adv. neural inf. proces. syst.*, 2019, p. 4694–4703.

[72] A. Romero, N. Ballas, S. Kahou, A. Chassang, C. Gatta, and Y. Bengio, “Fitnet: hints for thin deep nets,” in *Proc. Int. Conf. Learn. Represent.*, 2015, pp. 1–13.

[73] J. Liu, D. Wen, H. Gao, W. Tao, T. Chen, K. Osa, and M. Kato, “Knowledge representing: efficient, sparse, representation of prior knowledge for knowledge distillation,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. Workshop (CVPRW)*, 2019, pp. 638–646.

[74] J. Kim, S. Park, and N. Kwak, “Paraphrasing complex network: network compression via factor transfer,” in *Proc. Conf. Neural Inf. Process. Syst.*, 2018, pp. 2765–2774.

[75] G. Zhou, Y. Fan, R. Cui, W. Bian, X. Zhu, and K. Gai, “Rocket launching: a universal and efficient framework for training well-performing light net,” in *Proc. AAAI Conf. Artif. Intell.*, 2018, pp. 4580–4587.

[76] Y. Guan, P. Zhao, B. Wang, Y. Zhang, C. Yao, K. Bian, and J. Tang, “Differentiable feature aggregation search for knowledge distillation,” in *Proc. Lect. Notes Comput. Sci., ECCV*, 2020, pp. 469–484.

[77] F. Tang and G. Mori, “Similarity-preserving knowledge distillation,” in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2019, pp. 1365–1374.

[78] L. Yu, V. Yazici, X. Liu, J. Weijer, Y. Chen, and A. Ramisa, “Learning metrics from teachers: compact networks for image embedding,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2019, pp. 2902–2911.

[79] Y. Liu, J. Gao, B. Li, C. Yuan, W. Hu, Y. Li, and Y. Duan, “Knowledge distillation via instance relationship graph,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2019, pp. 7089–7097.

[80] H. Chen, Y. Wang, C. Xu, C. Xu, and D. Tao, “Learning student networks via feature embedding,” *IEEE Trans. Neur. Net. Lear.*, vol. 32, no. 1, pp. 25–35, 2021.

[81] A. Bagnall, H. Dau, J. Lines, M. Flynn, J. Large, A. Bostrom, P. Southam, and E. Keogh, “The uea multivariate time series classification archive, 2018,” *arXiv preprint arXiv: 1811.00075*, 2018.

[82] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of deep bidirectional transformers for language understanding,” in *Proc. Conf. N. Am. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2019, pp. 4171–4186.

[83] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, “Language models are few-shot learners,” in *Proc. Adv. neural inf. proces. syst.*, 2020, p. 1877–1901.

[84] J. Yim, D. Joo, J. Bae, and J. Kim, “A gift from knowledge distillation: fast optimization, network, minimization and transfer learning,” in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2017, pp. 7130–7138.

[85] Z. Huang and N. Wang, “Like what you like: knowledge distill via neuron selectivity transfer,” *arXiv preprint arXiv: 1707.01219*, 2017.

[86] S. You, C. Yu, C. Xu, and D. Tao, “Learning from multiple teacher networks,” in *Proc. ACM SIGKDD Int. Conf. Knowl. Discov. Data Min.*, 2017, p. 1285–1294.

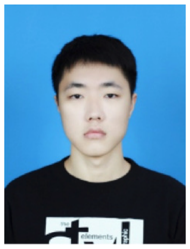
[87] N. Passalis, M. Tzelepi, and A. Tefas, “Probabilistic knowledge transfer for lightweight deep representation learning,” *IEEE Trans. Neur. Net. Lear.*, vol. 32, no. 5, pp. 2030–2039, 2021.

[88] S. Park and N. Kwak, “Feature-level ensemble knowledge distillation for aggregating knowledge from multiply networks,” in *Proc. Front. Artif. Intell. Appl.*, 2020, pp. 1–8.

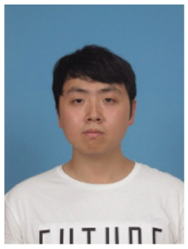
Zhiwen Xiao (M), received the B.Eng. degree in network engineering from the Chengdu University of Information Technology in 2019, Chengdu, China, and the M.Eng. degree in computer science from the Northwest A & F University in 2023, Yangling, China. He is pursuing the Ph.D. degree in computer science at Southwest Jiaotong University, Chengdu, China. His research interests include semantic communication, federated learning (FL), representation learning, data mining, and computer vision.

Xin Xu, received her B. Eng. degree in computer science from Central South University in 2016, and her M. Eng. degree in computer science from China University of Mining and Technology in 2021. Her interests lie in deep learning, computer version, data mining, video segmentation, and representation learning.

Huanlai Xing (M), received Ph.D. degree in computer science from University of Nottingham (Supervisor: Dr Rong Qu), Nottingham, U.K., in 2013. He was a Visiting Scholar in Computer Science, The University of Rhode Island (Supervisor: Dr. Haibo He), USA, in 2020-2021. Huanlai Xing is with the School of Computing and Artificial Intelligence, Southwest Jiaotong University (SWJTU), and Tangshan Institute of SWJTU. He was on Editorial Board of SCIENCE CHINA INFORMATION SCIENCES. He was a member of several international conference program and senior program committees, such as ECML-PKDD, MobiMedia, ISCIT, ICCC, TrustCom, IJCNN, and ICSINC. His research interests include semantic communication, representation learning, data mining, reinforcement learning, machine learning, network function virtualization, and software defined networking.



Bowen Zhao, received his B. Eng. degree in Computer Science and Technology in 2020, from Southwest Jiaotong University, Sichuan, China. He is currently pursuing the Ph.D. degree in the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. His research interests include deep reinforcement learning, cloud computing, and deep learning.



Xinhan Wang, received his B. Eng. degree in Computer Science and Technology in 2016, from Southwest University, Chongqing, China. He received his Ph.D. degree in Computer Science in 2023, from Southwest Jiaotong University, Chengdu, China. His research interests include network function virtualization, software defined networking, and evolutionary algorithm.

Fuhong Song, received the M.Eng. degree in computer technology and the Ph.D. degree in computer science and technology from Southwest Jiaotong University, Chengdu, China, in 2018 and 2022, respectively. He is currently a Lecturer with the School of Information, Guizhou University of Finance and Economics. His research interests include edge computing, multi-objective optimization and reinforcement learning.

Rong Qu (SM'12), is a full Professor at the School of Computer Science, University of Nottingham. She received her B.Sc. in Computer Science and Its Applications from Xidian University, China in 1996 and Ph.D. in Computer Science from The University of Nottingham, U.K. in 2003. Her research interests include the modelling and optimisation for logistics transport scheduling, personnel scheduling, network routing, portfolio optimization and timetabling problems by using evolutionary algorithms, mathematical programming, constraint programming in operational research and artificial intelligence. These computational techniques are integrated with knowledge discovery, machine learning and data mining to provide intelligent decision support on logistic fleet operations at SMEs, workforce scheduling at hospitals, policy making in education, and cybersecurity for connected and autonomous vehicles.

Dr. Qu is an associated editor at Engineering Applications of Artificial Intelligence, IEEE Computational Intelligence Magazine, IEEE Transactions on Evolutionary Computation, Journal of Operational Research Society, and PeerJ Computer Science. She is a Senior IEEE Member since 2012 and the Vice-Chair of Evolutionary Computation Task Committee since 2019 and Technical Committee on Intelligent Systems Applications (2015-2018) at IEEE Computational Intelligence Society. She has guest edited special issues on the automated design of search algorithms and machine learning at the IEEE Transactions on Pattern Analysis and Machine Intelligence and IEEE Computational Intelligence Magazine.

Li Feng, received the Ph.D. degree in control science and engineering from the Xi'an Jiaotong University, Xi'an, China, in 2005, under the supervision of Prof. Xiaohong Guan (Academian of CAS, IEEE Fellow). He is a Research Professor and PhD supervisor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu. His research interests include artificial intelligence, cyber security and its applications.