Densely Knowledge-aware Network for Multivariate Time Series Classification

Zhiwen Xiao, *Member, IEEE*, Huanlai Xing, *Member, IEEE*, Rong Qu, *Senior Member, IEEE*, Li Feng, Shouxi Luo, *Member, IEEE*, Penglin Dai, *Member, IEEE*, Bowen Zhao, and Yuanshun Dai, *Member, IEEE*

***Abstract*—Multivariate time series classification (MTSC) based on deep learning (DL) has attracted increasingly more research attention. The performance of a DL-based MTSC algorithm is heavily dependent on the quality of the learned representations providing semantic information for downstream tasks, e.g., classification. Hence, a model's representation learning ability is critical for enhancing its performance. This paper proposes a densely knowledge-aware network (DKN) for MTSC. The DKN's feature extractor consists of a residual multi-head convolutional network (ResMulti) and a transformer-based network (Trans), called ResMulti-Trans. ResMulti has five residual multi-head blocks for capturing the local patterns of data while Trans has three transformer blocks for extracting the global patterns of data. Besides, to enable dense mutual supervision between lower- and higher-level semantic information, this paper adapts densely dual self-distillation (DDSD) for mining rich regularizations and relationships hidden in the data. Experimental results show that compared with 5 state-of-the-art self-distillation variants, the proposed DDSD obtains 13/4/13 in terms of 'win'/'tie'/'lose' and gains the lowest AVG_rank score. In particular, compared with pure ResMulti-Trans, DKN results in 20/1/9 regarding 'win'/'tie'/'lose'. Last but not least, DKN overweighs 18 existing MTSC algorithms on 10 UEA2018 datasets and achieves the lowest AVG_rank score.**

***Index Terms*—Data Mining, Deep Learning, Knowledge Distillation, Multivariate Time Series Classification, Transformer**

I. Introduction

MULTIVARIATE time series data has been seen in various domains, such as electroencephalogram (EEG) analysis [1], [2], fault diagnosis [3], electrocardiogram (ECG) identification [4], anomaly detection [5], and mental health service [6]. Unlike other data, e.g., ImageNet [1] for image classification, Stanford Sentiment Treebank (SST-2) [2] for sentiment classification, and UCF101 [3] for video classification, multivariate time series is a sequence of timely ordered data points associated with multiple time-dependent variables that contain both local and global patterns. A multivariate time series classification (MTSC) algorithm is responsible for capturing the local and global patterns from each univariate time series (UTS) and discovering the connections among these UTS sequences, simultaneously [7].

Recently, deep learning (DL) based algorithms have attracted extensive attention in the MTSC community. By accurately modeling the internal data representation hierarchy, these algorithms can reflect the inherent connections among representations [7], [8], [9]. DL-based MTSC algorithms can be roughly divided into two streams: single-network-based and dual-network-based. A single-network-based model adopts a single (usually hybridized) network structure for feature and relation extraction. For example, Lee *et al.* [10] introduced a dynamic temporal pooling network to extract high-level features. Ma *et al.* [11] proposed an end-to-end adversarial joint-learning recurrent neural network (AJ-RNN) for feature extraction. Chen *et al.* [12] designed a dual-attention network to discover local and global patterns hidden in data. On the contrary, a dual-network-based model is usually composed of two parallel networks, one for local feature extraction and the other for global relation capture. Convolutional neural networks (CNNs) are generally adopted for extracting local features, while recurrent neural networks (RNNs)- and attention-based networks are usually used for capturing the connections among the features extracted. For example, a robust temporal feature network (RTFN) containing a temporal feature network and a long short-term memory (LSTM)-based attention network (LSTMaN) was used for supervised classification and unsupervised clustering [13]. An LSTM-fully convolutional network (LSTM-FCN) that combined FCN and LSTM-based networks in parallel was applied to MTSC [14]. However, most single- and dual-network-based MTSC models above lack in-depth self-reflection on their structures, restricting their ability for representation learning.

Within a representation hierarchy, the quality of the semantic information learned from lower and higher levels significantly affects a model's performance [15]. As known, higher-level semantic information is learned from lower-level semantic information. On the other hand, almost all the existing models update their parameters by the backpropagation (BP) method [16]. Lower-level semantic information is, to a

Manuscript received XXX; accepted XX XXX 2023. 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, L. Feng, S. Luo, P. Dai, B. Zhao, and Y. Dai are with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 610031, China, with the Tangshan Institute of Southwest Jiaotong University, Tangshan 063000, China, and with the Engineering Research Center of Sustainable Urban Intelligent Transportation, Ministry of Education, China (Emails: xiao1994zw@163.com; hxx@home.swjtu.edu.cn; fengli@swjtu.edu.cn; sxluo@swjtu.edu.cn; penglindai@swjtu.edu.cn; cn16bz@icloud.com; 1125105129@qq.com).

R. Qu is with the School of Computer Science, University of Nottingham, Nottingham NG8 1BB, UK (Email: rong.qu@nottingham.ac.uk)

[1]https://image-net.org/

[2]https://nlp.stanford.edu/sentiment/code.html

[3]http://crcv.ucf.edu/papers/UCF101_CRCV-TR-12-01.pdf

certain extent, affected by higher-level semantic information. *Thus, lower- and higher-level semantic information learns from and influences each other during the model learning process. Efficiently promoting mutual learning between lower and higher levels, seems a promising solution to enhancing a model's performance during learning.*

Recently, self-distillation has become one of the main streams in knowledge distillation (KD). A self-distillation-based model is a teacher and its own student, promoting knowledge flow within the model [17]. For example, Zhang *et al.* [18] presented an efficient self-distillation method called Be Your Own Teacher (BYOT) to transfer the output's knowledge to each lower-level module. Ji *et al.* [19] developed a self-distillation refine approach to promote knowledge flow from higher to lower levels and enhance the model's classification accuracy. Representative algorithms also include layer-wise attention self-distillation [20], ensemble self-distillation [21], transitive self-distillation [21], end-to-end progressive self-label correction (ProSelfLC) [22]. However, almost all the existing self-distillation algorithms emphasized knowledge transfer from higher to lower levels, ignoring the significance of lower-level semantic information to higher-level semantic information.

To address the problem above, we propose a densely knowledge-aware network (DKN) for MTSC. Unlike unidirectional self-distillation methods that enable knowledge transfer from higher to lower levels, e.g., BYOT and ensemble self-distillation, DKN adopts densely dual self-distillation (DDSD) to offer dense mutual learning between lower and higher levels, efficiently enhancing its representation regularization ability.

Our major contributions are summarized below.

- This paper designs a dual-network-based feature extractor for DKN, namely ResMulti-Trans, where a residual multi-head convolutional network (ResMulti) and a transformer-based network (Trans) lie in parallel, as shown in Fig. 1. ResMulti with five residual multi-head blocks and Trans with three transformer blocks are used for local and global pattern extraction, respectively.
- Through the DDSD, this paper enables dense mutual supervision between lower- and higher-level semantic information, helping DKN mine rich regularizations and relationships hidden in the data.
- Experiments show that DKN outperforms 18 existing MTSC algorithms regarding the 'win'/'tie'/'lose' measure and AVG_rank, where results are based on the top-1 accuracy. Specifically, DKN wins 10 out of 30 datasets and achieves the lowest AVG_rank score, namely 5.550. Our DDSD is better than 5 state-of-the-art self-distillation variants since it achieves 13/4/13 in terms of 'win'/'tie'/'lose' and obtains the lowest AVG_rank score, namely 2.250. DKN beats the pure ResMulti-Trans on 21 datasets regarding the top-1 accuracy.

The remainder of the paper is summarized as follow. Section II reviews a number of existing MTSC algorithms. Section III overviews the DKN's structure and introduces its key components. The experimental analysis and conclusion are provided and summarized in Sections IV and V, respectively.

II. Related Work

This section reviews some traditional and DL-based MTSC algorithms.

A. Traditional Algorithms

Distance- and feature-based algorithms are two main research streams for MTSC [7], [12]. Integrating the nearest neighbor (NN) and dynamic time warping (DTW) is distance-based, measuring the similarities between spatial features of data, e.g., DTW_A, DTW_I, and DTW_D [23]. A large number of DTW-NN-based ensemble algorithms have been developed for MTSC, e.g, the elastic ensemble (EE) with 11 1-NN-based elastic distance [24], transformation-based ensemble (COTE) with 37 NN-based classifiers [25], hierarchical vote collective of transformation-based ensembles (HIVE-COTE) [26], random interval spectral ensemble (RISE) [26], explainable-by-design ensemble method (XEM) [27], and HIVE-COTE 2.0 [28].

Feature-based algorithms focus on capturing the representative features from input data. For example, Baydogan and Runger [29] introduced a pattern-based representation method called learned pattern similarity (LPS) for feature extraction. Shifaz *et al.* [30] proposed a scalable and accurate forest algorithm for addressing MTSC problems. Baldán and Benítez [31] presented an alternative representation method to improve the interpretability of time series. Typical feature-based algorithms also include the time series forest (TSF) [32], hidden-unit logistic model (HULM) [33], bag-of-features structure [34], bag of symbolic Fourier approximation symbols (BOSS) [35], Contractable BOSS (CBOSS) [35], online rule-based classifier learning [36], active semi-supervised learning [37], autoregressive tree-based ensemble approach (mv-ARF) [38], fuzzy cognitive map [39], and WEASEL+MUSE [40].

B. DL-based Algorithms

DL-based MTSC algorithms are good at modeling an internal data representation hierarchy, focusing on the inherent relationships among representations [7]. Single- and dual-network-based models present research streams [8], [9]. The dynamic temporal pooling network [10], AJ-RNN [11], DA-Net [12], InceptionTime [41], FCN [42], ResNet [42], multi-process collaborative architecture [43], ROCKET [44], shapelet-neural network [45], deep contrastive representation learning with self-distillation [46], MiniROCKET [47], echo state network [48] and reservoir computing [49] are widely recognized single-network-based models. The well-known RTFN [13], ResNet-Transformer [50], LSTM-FCN [14], SelfMatch [51], TapNet [52], and RNTS [53] are all dual-network-based.

III. The Proposed DKN

This section first describes the structure of DKN and its key components, including the residual multi-head block, transformer, and densely dual self-distillation (DDSD). Then, it introduces the loss function.

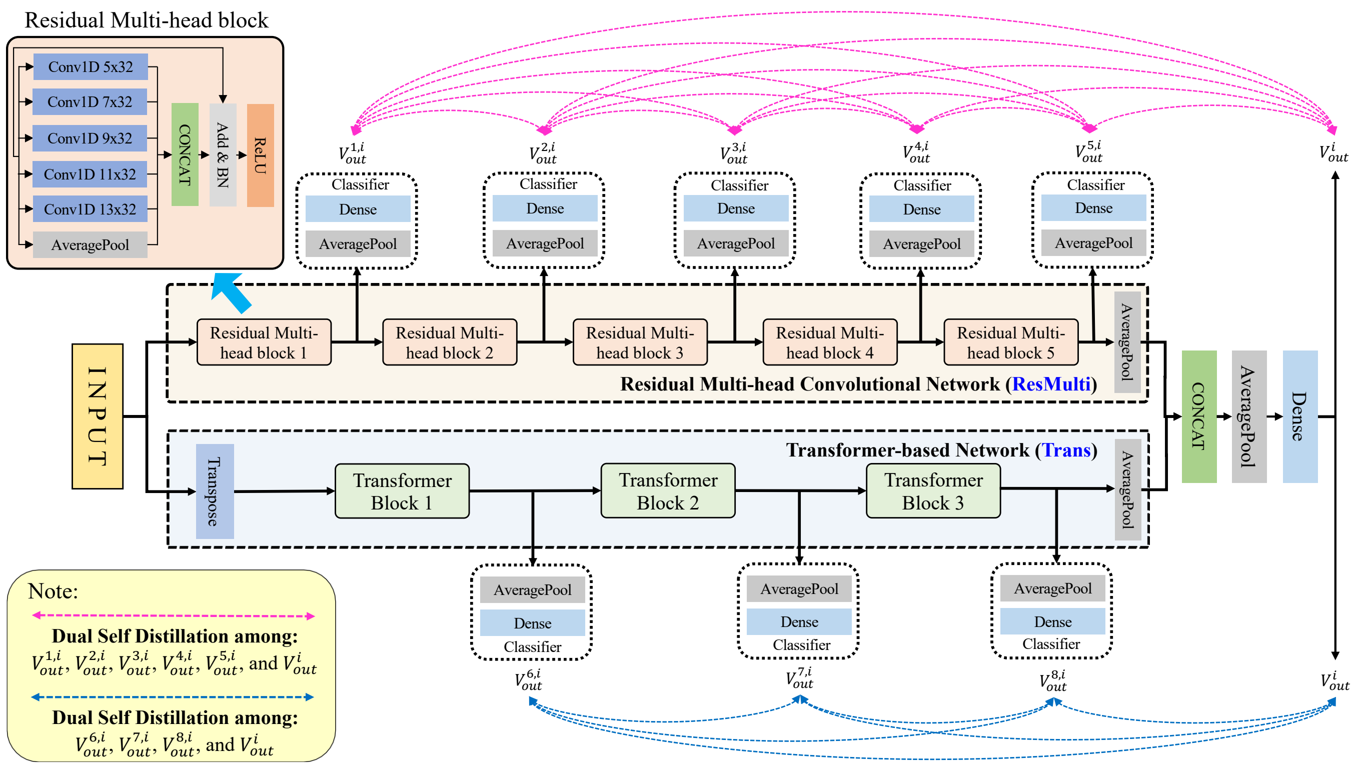

Residual Multi-head block
Conv1D 5x32
Conv1D 7x32
Conv1D 9x32
Conv1D 11x32
Conv1D 13x32
AveragePool
CONCAT
Add & BN
ReLU
INPUT
Residual Multi-head block 1
Residual Multi-head block 2
Residual Multi-head block 3
Residual Multi-head block 4
Residual Multi-head block 5
AveragePool
Residual Multi-head Convolutional Network (ResMulti)
Transformer-based Network (Trans)
Transpose
Transformer Block 1
Transformer Block 2
Transformer Block 3
Classifier
Dense
CONCAT
AveragePool
Dense
$V_{out}^{1,i}$ $V_{out}^{2,i}$ $V_{out}^{3,i}$ $V_{out}^{4,i}$ $V_{out}^{5,i}$ V_{out}^{i}
$V_{out}^{6,i}$ $V_{out}^{7,i}$ $V_{out}^{8,i}$ V_{out}^{i}
Note:
Dual Self Distillation among: $V_{out}^{1,i}$, $V_{out}^{2,i}$, $V_{out}^{3,i}$, $V_{out}^{4,i}$, $V_{out}^{5,i}$, and V_{out}^{i}
Dual Self Distillation among: $V_{out}^{6,i}$, $V_{out}^{7,i}$, $V_{out}^{8,i}$, and V_{out}^{i}

Fig. 1. The overview of DKN. The feature extractor, called ResMulti-Trans, consists of ResMulti and Trans. ResMulti has five residual multi-head blocks and one average pooling layer. Each multi-head block primarily contains five one-dimensional convolutional neural network (Conv1D) modules and one average pooling layer, where "Conv1D 9x32" denotes a Conv1D with a kernel size of 9 and a channel number of 32. Trans contains three transformer blocks and one average pooling layer. Note: "Transpose" outputs the transpose of a given matrix and "BN" is the batch normalization module. Let $V_{out}^{j,i}$, $i = 1, 2, ..., N$, $j = 1, 2, 3, 4, 5$, denote the output of the i-th output feature vector of j-th multi-head block after passing the corresponding classifier in ResMulti, where N is the size of input samples. Let $V_{out}^{j,i}$, $i = 1, 2, ..., N$, $j = 6, 7, 8$, be the output of the i-th output feature vector of j-th transformer block after passing the corresponding classifier in Trans. V_{out}^{i} represents the i-th output feature vector of DKN.

A. Overview

DKN aims to effectively promote the mutual flow between lower- and higher-level semantic information, extracting rich regularizations and relationships hidden in the data. The structure of DKN is illustrated in Fig. 1. ResMulti-Trans is the extractor of local features and global relations. With five residual multi-head blocks and one average pooling layer, ResMulti focuses on extracting local patterns of data, while Trans, with three transformer blocks and one average pooling layer, is responsible for discovering global patterns of data. In addition, DDSD is adopted to provide dense mutual supervision between lower- and higher-level semantic information, which enhances the DKN model's representation learning ability.

B. Residual Multi-head Block

In ResMulti, the residual multi-head blocks are used to capture multi-scale local features from the data. To be specific, each block mainly consists of five one-dimensional convolutional neural network (Conv1D) modules and one average pooling module, as shown in Fig. 1. Note that the five Conv1D modules are "Conv1D 5x32", "Conv1D 7x32", "Conv1D 9x32", "Conv1D 11x32", and "Conv1D 13x32". Note that "Conv1D 5x32" represents a Conv1D with a kernel size of 5 and a channel number of 32. An arbitrary Conv1D module is defined as:

$$f_{conv}(x) = W_{conv} \otimes x + b_{conv} \tag{1}$$

where, x stands for the input data. W_{conv} and b_{conv} are the weight and bias matrices of Conv1D, respectively.

The residual structure is adopted to avoid the loss of necessary information and gradient degradation during training. Let V_{cnn1}, V_{cnn2}, V_{cnn3}, V_{cnn4}, V_{cnn5}, and V_{cnn6} denote the outputs of "Conv1D 5x32", "Conv1D 7x32", "Conv1D 9x32", "Conv1D 11x32", "Conv1D 13x32", and the average pooling module, respectively. For an arbitrary residual multi-head block, its output, V_{ResM}, is defined in Eq. (2).

$$V_{ResM} = f_{ReLU}(f_{BN}(f_{concat}([V_{cnn1}, ..., V_{cnn6}])) + x) \tag{2}$$

where, f_{ReLU}, f_{BN}, and f_{concat} are the rectified linear unit activation (ReLU), batch normalization (BN), and CONCAT functions, respectively.

C. Transformer Block

In Trans, the three transformer blocks are responsible for capturing global pattens from the data, where each block relates the features at different locations of its input [54]. The architecture of a transformer block is shown in Fig. 2.

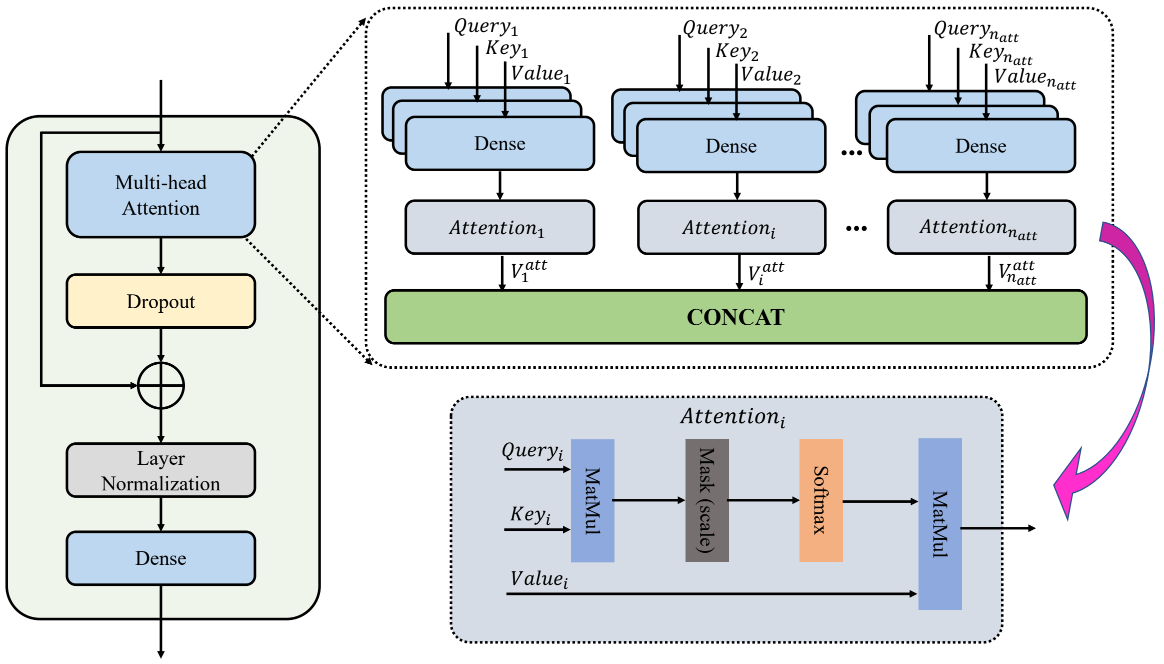


Fig. 2. Architecture of transformer [54] block. Note: "MatMul" is the matrix multiplication operation.

There are n_{att} attention modules in the multi-head attention. The i-th attention module, $Attention_i$, maps a query, $Query_i$, and a set of key-value pairs, $Key_i - Value_i$, to an output, V_i^{att}. V_i^{att} is defined as:

$$V_i^{att} = f_{softmax}(\frac{Query_i \cdot Key_i^T}{\sqrt{d_i}}) \cdot Value_i \tag{3}$$

where, Key_i^T and d_i are the transpose and dimension of Key_i, respectively. $f_{softmax}$ computes the possibilities of a give matrix.

D. Densely Dual Self-distillation (DDSD)

The DDSD technique promotes mutual knowledge transfer between lower- and higher-level semantic information, which helps regularize the model and improve its representation learning performance. Its structure is shown in Fig. 1. Let F_i^1, F_i^2, F_i^3, F_i^4 and F_i^5 denote the i-th (i = 1, ..., N) output feature vectors of the five residual multi-head blocks in ResMulti, where N is the number of input samples. Let F_i^6, F_i^7, and F_i^8 be the i-th output feature vectors of the three transformer blocks in Trans. As suggested in [17], [18], [19], [20], [21], to match each vector with the output vector of DKN, V_{out}^i, we associate each feature vector with a specific classifier consisting of one average pooling layer and one dense module, denoted by f_{class}. The dense module in classifier f_{class} has C neurons, where C is the number of classes. For an arbitrary feature vector, F_i^j, $j = 1, ..., 8$, its output after passing the corresponding classifier, $V_{out}^{j,i}$, is defined as:

$$V_{out}^{j,i} = f_{softmax}(f_{class}(F_i^j)/T) \quad i = 1, ..., N, j = 1, ..., 8 \tag{4}$$

where, T is a temperature scaling parameter. In this paper, we set T = 1.0 (see Section IV-C).

The loss function of DDSD, $\mathcal{L}_{KD}$, is calculated as:

$$\begin{aligned}\mathcal{L}_{KD} = \frac{1}{N}\sum_{i=1}^{N}(&\sum_{j=1}^{8}(f_{KL}(V_{out}^i, V_{out}^{j,i}) + f_{KL}(V_{out}^{j,i}, V_{out}^i))\\ &+\sum_{k=2}^{5}\sum_{j=1}^{k}(f_{KL}(V_{out}^{k,i}, V_{out}^{j,i}) + f_{KL}(V_{out}^{j,i}, V_{out}^{k,i}))\\ &+\sum_{k=6}^{8}\sum_{j=5}^{k}(f_{KL}(V_{out}^{k,i}, V_{out}^{j,i}) + f_{KL}(V_{out}^{j,i}, V_{out}^{k,i})))\end{aligned} \tag{5}$$

where, f_{KL} is the Kullback–Leibler (KL) function.

E. Loss Function

The loss function of DKN, $\mathcal{L}$, consists of a supervised loss, $\mathcal{L}_{sup}$, and a DDSD loss, $\mathcal{L}_{KD}$. Like the previous studies in [17], [18], [19], [20], [21], $\mathcal{L}_{sup}$ is based on the cross-entropy function that calculates the differences between the ground-truth labels and their prediction vectors, as written in Eq. (6).

$$\mathcal{L}_{sup} = -\frac{1}{N}\sum_{i=1}^{N} y_i log(V_{out}^i) \tag{6}$$

where, y_i is the i-th ground truth label.

The loss function of DKN, $\mathcal{L}$, is defined in Eq. (7).

$$\mathcal{L} = \mu\mathcal{L}_{sup} + (1-\mu)\mathcal{L}_{KD} + \epsilon||\theta||_2^2 \tag{7}$$

where, μ is a coefficient reflecting the relative importance of $\mathcal{L}_{sup}$ over $\mathcal{L}_{KD}$. In this paper, we set μ = 0.9 (more details are found in Section IV-C). θ is the parameters of DKN. ϵ represents the coefficient of $||\theta||_2^2$ (L_2 regularization). Following [13], [51], [53], we set ϵ = 0.0005. The pseudo code of DKN is shown in Algorithm 1.

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

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

Algorithm 1 Procedure of DKN

```
Input: $\mathcal{D} = (\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test})$;   ▷ $\mathcal{D}_{train}, \mathcal{D}_{val}$ and $\mathcal{D}_{test}$ are the training, validation, and testing data sets, respectively.
Output: $\mathcal{Y}$;
1:  Initialize the DKN's parameters, $\theta_0$;
2:  for $i = 1$ to $Epochs$ do   ▷ $Epochs$ denotes the number of training epochs.
3:      Feedforward $\mathcal{D}_{train}$ into DKN;
4:      Obtain $\mathcal{L}_{KD}$ by Eq. (5);
5:      Obtain $\mathcal{L}_{sup}$ by Eq. (6);
6:      Obtain the DKN's loss, $\mathcal{L}$, by Eq. (7);
7:      Update $\theta_i$ by $\theta_i = \theta_{i-1} - \eta\nabla_{\theta_{i-1}}\mathcal{L}(\theta_{i-1})$;   ▷ $\eta$ represents the learning rate. $\nabla_{\theta_{i-1}}$ and $\theta_{i-1}$ are the DKN's parameters and gradient at the ($i$-1)-th training epoch, respectively.
8:      if $i > 1$ then
9:          Validate DKN based on $\mathcal{D}_{val}$;
10:     end if
11: end for
12: Predict $\mathcal{Y}$ based on $\mathcal{D}_{test}$.
```

IV. PERFORMANCE AND EVALUATION

This section first introduces the experimental setup, performance metrics, hyper-parameter sensitivity, and ablation study. Then, the DKN's performance and efficiency are verified. Finally, the case study is explained.

TABLE II
HYPER-PARAMETER SETTINGS OF THE THREE TRANSFORMER BLOCKS.

Transformer No.	n_{att}	Dense Layer's units	Dropout Value
1	8	64	0.5
2	8	128	0.5
3	8	192	0.5

A. Experimental Setup

1) Dataset Description: As the previous studies [12], [13], [14] suggested, we adopt the University of East Anglia multivariate time series archive in 2018 (UEA2018) [55] for algorithmic performance evaluation. UEA2018, a widely used MTSC archive, consists of 30 datasets in 7 application scenarios, including audio spectra, human activity recognition, electroencephalogram, meagnetoencephalography, motion, electrocardiogram, and others. More details are seen in Table I.

2) Implementation Details: The hyper-parameter settings of the three transformer blocks are shown in Table II. In this paper, we adopt the Adam optimizer with its initial learning rate, momentum term, decay value set to 0.001, 0.9, 0.9, respectively. We run the experiments using a computer with Ubuntu 18.04 OS, Python 3.7, an Nvidia GTX 1080Ti GPU with 11GB, Tensorflow 1.18, and an AMD R5 1400 CPU with 16G RAM.

B. Performance Metrics

To verify the proposed DKN, we consider two commonly used metrics, namely, 'win'/'tie'/'lose' and AVG_rank, which are based on the top-1 accuracy. As suggested in [7], [8],

TABLE III
THE TOP-1 ACCURACY RESULTS WITH DIFFERENT T VALUES ON 30 UEA2018 DATASETS.

Dataset Index	T					
	0.5	1.0	2.0	3.0	4.0	5.0
AWR	0.980	**0.993**	0.970	0.963	0.953	0.987
AF	0.400	**0.467**	**0.467**	0.400	0.333	0.267
BM	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CT	**0.986**	**0.986**	0.969	0.964	0.979	0.983
CK	0.972	0.951	0.958	0.972	**0.986**	**0.986**
DDG	0.500	0.560	0.520	0.480	**0.600**	0.580
EW	0.550	**0.628**	0.517	0.618	0.550	0.527
EP	0.964	0.979	0.978	0.964	**0.986**	0.978
EC	0.304	**0.372**	0.323	0.293	**0.372**	0.316
ER	0.919	**0.933**	0.919	0.919	0.919	0.922
FD	0.629	**0.631**	0.614	0.629	0.629	0.629
FM	0.590	**0.600**	0.580	0.580	0.580	0.570
HMD	0.608	**0.662**	0.541	0.500	0.544	0.544
HW	**0.231**	**0.231**	**0.231**	**0.231**	**0.231**	**0.231**
HB	0.717	**0.765**	0.717	0.727	0.751	0.751
IW	0.340	**0.362**	0.360	0.350	0.352	0.360
JV	**0.930**	**0.930**	**0.930**	**0.930**	**0.930**	**0.930**
LIB	0.894	**0.900**	0.883	0.894	0.883	0.883
LSST	**0.391**	0.347	0.257	0.333	0.333	0.333
MI	0.590	**0.620**	0.610	0.600	0.590	0.580
NATO	0.850	0.872	**0.883**	**0.883**	0.850	0.850
PD	0.939	**0.948**	0.939	0.939	0.939	0.911
PEMS	0.745	**0.930**	0.913	0.913	0.913	0.913
PS	0.421	**0.525**	0.421	0.425	0.421	0.421
RS	0.868	**0.879**	0.868	0.868	0.868	0.868
SRS1	0.908	**0.913**	0.899	0.899	0.908	0.908
SRS2	0.550	0.600	0.533	0.550	**0.611**	0.533
SAD	**0.946**	**0.963**	**0.963**	**0.963**	**0.963**	**0.963**
SWJ	0.400	**0.533**	0.500	0.500	0.500	**0.533**
UW	0.881	**0.897**	0.894	0.857	0.881	0.894
Win	1	**18**	0	0	2	0
Tie	5	6	6	5	**7**	6
Lose	24	**6**	24	25	21	24
Best	6	**24**	6	5	9	6
AVG_rank	3.950	**1.800**	4.017	4.017	3.517	3.700

TABLE IV
THE TOP-1 ACCURACY RESULTS WITH DIFFERENT μ VALUES ON 30 UEA2018 DATASETS.

Dataset Index	μ								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
AWR	0.953	0.953	0.970	0.963	0.987	0.987	0.990	0.990	**0.993**
AF	0.267	0.267	0.267	0.267	0.333	0.333	0.400	0.400	**0.467**
BM	0.975	0.975	0.975	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CT	0.917	0.931	0.931	0.931	0.969	0.969	0.983	0.983	**0.986**
CK	0.861	0.861	0.861	0.861	0.917	0.917	0.931	0.944	**0.951**
DDG	0.375	0.375	0.375	0.380	0.380	0.480	0.540	**0.560**	**0.560**
EW	0.511	0.511	0.511	0.527	0.527	0.550	0.618	0.626	**0.628**
EP	0.732	0.732	0.732	0.863	0.863	0.964	0.964	**0.979**	**0.979**
EC	0.293	0.293	0.323	0.316	0.323	0.323	**0.373**	0.372	0.372
ER	0.859	0.881	0.881	0.881	0.919	0.919	0.933	**0.941**	0.933
FD	0.519	0.529	0.545	0.513	0.555	**0.640**	0.629	0.629	0.631
FM	0.520	0.530	0.530	0.530	0.540	0.580	0.590	**0.620**	0.600
HMD	0.378	0.378	0.541	0.541	0.508	0.556	0.649	0.649	**0.662**
HW	0.191	0.191	0.191	0.191	0.191	**0.231**	**0.231**	**0.231**	**0.231**
HB	0.564	0.564	0.564	0.619	0.658	0.727	0.717	0.717	**0.765**
IW	0.228	0.237	0.237	0.237	0.237	**0.362**	**0.362**	**0.362**	**0.362**
JV	0.778	0.778	0.778	0.800	0.800	0.900	**0.968**	**0.968**	0.930
LIB	0.833	0.833	0.833	0.850	0.850	0.870	0.870	0.894	**0.900**
LSST	0.161	0.265	0.265	0.265	0.265	0.285	0.314	**0.352**	0.347
MI	0.500	0.500	0.500	0.520	0.540	0.560	0.560	0.600	**0.620**
NATO	0.800	0.839	0.839	0.839	0.839	0.850	0.850	**0.872**	**0.872**
PD	0.892	0.892	0.892	0.892	0.939	0.939	**0.951**	0.939	0.948
PEMS	0.734	0.734	0.734	0.745	0.745	0.914	0.914	**0.930**	**0.930**
PS	0.388	0.388	0.388	0.388	0.288	0.369	0.404	0.418	**0.525**
RS	0.803	0.803	0.803	0.842	0.842	0.868	0.868	0.854	**0.879**
SRS1	0.829	0.829	0.829	0.829	0.839	0.840	0.908	0.908	0.913
SRS2	0.483	0.483	0.483	0.483	0.533	0.533	0.533	**0.600**	**0.600**
SAD	0.787	0.787	0.787	0.900	0.787	0.900	0.900	**0.963**	**0.963**
SWJ	0.333	0.333	0.333	0.333	0.333	0.500	**0.533**	**0.533**	**0.533**
UW	0.868	0.868	0.868	0.869	0.881	0.881	**0.897**	**0.897**	**0.897**
Win	0	0	0	0	0	1	2	3	**8**
Tie	0	0	0	1	1	2	6	12	**14**
Lose	30	30	30	29	29	27	22	15	**8**
Best	0	0	0	1	1	3	8	15	**22**
AVG_rank	8.017	7.533	7.200	6.350	5.367	3.883	2.800	2.200	**1.650**

[10], [11], [12], [13], [14], for an arbitrary MTSC algorithm, its 'win', 'tie', and 'lose' scores reflect on how many datasets this algorithm is better than, equal to, and worse than the other algorithms for performance comparison, respectively; its 'best' score is the summation of the corresponding 'win' and 'tie' scores. Like the previous studies in [7], [8], [10], [13], [14], [50], [51], [52], [53], we use AVG_rank to differentiate various algorithms, where results are based on the Wilcoxon signed-rank test with Holm's alpha (5%) correction.

C. Hyper-parameter Sensitivity

We investigate the impact of hyper-parameter settings on the DKN's performance on 30 UEA2018 datasets.

1) DKN with different T values: T is the temperature scaling parameter that controls a soft probability distribution over classes. Table III shows the top-1 accuracy results obtained by DKN with different T values on 30 datasets. One can easily find that 1.0 helps DKN achieve the best 'win'/'tie'/'lose' result, namely 18/6/6, and the best AVG_rank value, namely 1.800. That is why we hereafter set T = 1.0 in the experiments.

2) DKN with different μ values: μ is the coefficient that balances between $\mathcal{L}_{sup}$ and $\mathcal{L}_{KD}$, resulting in low entropy during training. Table IV shows the top-1 accuracy results obtained by DKN with different T values on 30 datasets. It is seen that μ = 0.9 corresponds to the best 'win'/'tie'/'lose' result, namely 8/14/8, and the best AVG_rank value, namely 1.650. That reflects μ = 0.9 is beneficial to the entropy reduction of DKN.

3) ResMulti with different multi-head blocks: To study the effectiveness of different multi-head blocks on ResMulti, we compare ResMulti with five variants:

- ResMulti-(1): ResMulti with only one multi-head block.
- ResMulti-(2): ResMulti with two multi-head blocks.
- ResMulti-(3): ResMulti with three multi-head blocks.
- ResMulti-(4): ResMulti with four multi-head blocks.
- ResMulti: ResMulti with five multi-head blocks.
- ResMulti-(6): ResMulti with six multi-head blocks.

As shown in Table V, as the number of multi-head blocks increases, the accuracy of ResMulti becomes higher and higher. ResMulti with multiple multi-head blocks makes it easier to mine plenty of multi-scale local features from the input, e.g., ResMulti-(6) outperforms ResMulti-(1), ResMulti-(2), ResMulti-(3), ResMulti-(4), and ResMulti on the InsectWingbeat dataset.

ResMulti and ResMulti-(6) obtain the same results on 23 datasets. Meanwhile, the mean accuracy of ResMulti-(6) is only 0.001 higher than that of ResMulti, reflecting that ResMulti and ResMulti-(6) have similar performance to some extent. Compared with ResMulti, ResMulti-(6) consumes more computational resources, e.g., the parameters of ResMulti and ResMulti-(6) on the EigenWorms dataset are 360,942 and 361,135, respectively. This is why ResMulti uses five multi-head blocks rather than six.

4) Trans with different transformer blocks: To investigate the effectiveness of different transformer blocks on Trans, we compare the proposed Trans with three variants:

- Trans-(1): ResMulti with only one transformer block.

TABLE V
THE TOP-1 ACCURACY RESULTS OBTAINED BY VARIOUS RESMULTI VARIANTS ON 30 UEA2018 DATASETS.

Dataset Index	ResMulti-(1)	ResMulti-(2)	ResMulti-(3)	ResMulti-(4)	ResMulti	ResMulti-(6)
AWR	0.884	0.905	0.934	0.953	0.969	**0.973**
AF	0.200	0.300	0.300	**0.400**	**0.400**	**0.400**
BM	0.700	0.900	**1.000**	**1.000**	**1.000**	**1.000**
CT	0.782	0.814	0.868	**0.969**	**0.969**	**0.969**
CK	0.705	0.778	0.827	0.848	**0.903**	**0.903**
DDG	0.100	0.200	0.200	0.260	**0.320**	**0.320**
EW	0.208	0.317	0.389	0.422	0.518	**0.527**
EP	0.500	0.528	0.739	0.905	**0.920**	**0.920**
EC	0.075	0.203	0.275	0.293	**0.316**	**0.316**
ER	0.133	0.133	0.133	0.881	**0.881**	**0.881**
FD	0.136	0.238	0.378	0.549	**0.555**	**0.555**
FM	0.200	0.320	0.360	0.560	**0.580**	**0.580**
HMD	0.270	0.284	0.324	0.378	**0.541**	**0.541**
HW	0.080	0.100	0.120	0.160	0.184	**0.191**
HB	0.199	0.218	0.360	0.502	**0.619**	**0.619**
IW	0.008	0.025	0.125	0.200	0.227	**0.228**
JV	0.458	0.693	0.772	0.817	**0.916**	**0.916**
LIB	0.500	0.700	0.727	0.800	**0.833**	**0.833**
LSST	0.243	0.275	0.305	0.390	**0.408**	0.391
MI	0.300	0.380	0.420	0.550	**0.570**	**0.570**
NATO	0.400	0.600	0.750	0.800	**0.850**	**0.850**
PD	0.500	0.583	0.725	0.826	**0.892**	**0.892**
PEMS	0.485	0.522	0.697	0.745	**0.745**	**0.745**
PS	0.104	0.104	0.104	0.151	**0.288**	**0.288**
RS	0.652	0.652	0.741	0.842	0.842	**0.854**
SRS1	0.458	0.582	0.696	0.804	**0.867**	**0.867**
SRS2	0.300	0.450	0.483	0.494	**0.550**	**0.550**
SAD	0.100	0.758	0.805	0.883	0.939	**0.946**
SWJ	0.267	0.333	0.333	0.400	**0.400**	**0.400**
UW	0.500	0.683	0.833	0.833	**0.881**	**0.881**
Win	0	0	0	0	1	**6**
Tie	0	0	1	3	**23**	**23**
Lose	30	30	29	27	6	**1**
Best	0	0	1	3	24	**29**
Mean Accuracy	0.348	0.453	0.524	0.621	0.663	**0.664**

TABLE VI
THE TOP-1 ACCURACY RESULTS OBTAINED BY VARIOUS TRANS VARIANTS ON 30 UEA2018 DATASETS.

Dataset Index	Trans-(1)	Trans-(2)	Trans	Trans-(4)
AWR	0.736	0.884	0.953	**0.957**
AF	0.167	0.267	**0.333**	**0.333**
BM	0.676	**1.000**	**1.000**	**1.000**
CT	0.814	0.917	**0.931**	**0.931**
CK	0.668	0.827	0.889	**0.903**
DDG	0.100	0.200	**0.380**	**0.380**
EW	0.275	0.330	**0.511**	**0.511**
EP	0.500	0.666	0.732	**0.883**
EC	0.106	0.203	0.304	**0.316**
ER	0.133	0.133	0.859	**0.874**
FD	0.238	0.378	**0.573**	**0.573**
FM	0.200	0.400	0.550	**0.560**
HMD	0.270	0.324	**0.508**	**0.508**
HW	0.120	0.153	**0.191**	**0.191**
HB	0.218	0.502	**0.658**	**0.658**
IW	0.125	0.200	**0.237**	**0.237**
JV	0.458	0.693	0.768	**0.817**
LIB	0.400	0.727	**0.806**	**0.806**
LSST	0.104	0.226	**0.265**	**0.265**
MI	0.300	0.480	**0.550**	**0.550**
NATO	0.300	0.600	**0.800**	**0.800**
PD	0.538	0.725	**0.892**	**0.892**
PEMS	0.339	0.589	0.745	**0.751**
PS	0.104	0.151	**0.269**	**0.269**
RS	0.359	0.741	0.856	**0.867**
SRS1	0.652	0.771	**0.823**	**0.823**
SRS2	0.450	0.494	**0.533**	**0.533**
SAD	0.202	0.639	**0.787**	**0.787**
SWJ	0.200	0.333	**0.400**	**0.400**
UW	0.500	0.700	0.859	**0.869**
Win	0	0	0	**8**
Tie	0	1	**22**	**22**
Lose	30	29	8	**0**
Best	0	1	22	**30**
Mean Accuracy	0.342	0.508	0.632	**0.641**

- Trans-(2): ResMulti with two transformer blocks.
- Trans: Trans with three transformer blocks.
- Trans-(4): Trans with four transformer blocks.

Table VI shows the top-1 accuracy results obtained by various Trans variants on 30 datasets. First, Trans-(4) achieves the best performance, because the four transformer blocks capture more affluent global relations from the data.

Trans and Trans-(4) result in similar performance on 22 datasets. The average accuracy of Trans-(4) is slightly higher than that of Trans (about 0.009), showing that Trans and Trans-(4) have similar performance to some extent. Compared with Trans, Trans-(4) require more computational resources, e.g., the parameters of Trans and Trans-(4) on the EigenWorms dataset are 4,859,246 and 5,191,429, respectively. This is why we choose Trans with three transformer blocks rather than four.

D. Ablation Study

This section evaluates the key components of DKN on 30 UEA2018 datasets.

1) Effectiveness of ResMulti and Trans: To study the effectiveness of ResMulti and Trans, we compare the proposed DKN with two variants:

- DKN-w/o-Trans: DKN without the transformer-based network.
- DKN-w/o-ResMulti: DKN without the residual multi-head convolutional network.

Table VII shows the top-1 accuracy results obtained by DKN and its two variants on 30 datasets. That DKN outperforms DKN-w/o-Trans on 28 datasets (except BM and LSST) reflects the effectiveness of Trans, i.e., the three transformer blocks can mine sufficient global relations from a given input. That DKN overwhelms DKN-w/o-ResMulti on 29 datasets (except BM) indicates the effectiveness of ResMulti, namely, the five residual multi-head blocks can extract abundant multi-scale local features from the input. With Trans and ResMulti, DKN is able to capture more high-quality representations and thus obtains better performance with respect to top-1 accuracy.

2) Effectiveness of DDSD: To study the effectiveness of DDSD, we compare it with five existing self-distillation variants, including BYOT, SAD, TSD, ProSelfLC, and SelfRef. The models for performance comparison are listed below.

- ResMulti-Trans: DKN without DDSD, i.e., the pure ResMulti-Trans.
- BYOT-ResMulti-Trans: ResMulti-Trans with the best teacher distillation instead of DDSD [18].
- SAD-ResMulti-Trans: ResMulti-Trans with the layer-wise attention self-distillation instead of DDSD [20].
- TSD-ResMulti-Trans: ResMulti-Trans with the transitive self-distillation instead of DDSD [21].
- ProSelfLC-ResMulti-Trans: ResMulti-Trans with the progressive self-label correction instead of DDSD [22] .
- SelfRef-ResMulti-Trans: ResMulti-Trans with the self-distillation refine instead of DDSD [19].

The top-1 accuracy results with different DKN variants on 30

TABLE VII
THE TOP-1 ACCURACY RESULTS OBTAINED BY VARIOUS DKN VARIANTS ON 30 UAE2018 DATASETS.

Dataset Index	DKN-w/o-Trans	DKN-w/o-ResMulti	ResMulti-Trans	BYOT-ResMulti-Trans	SAD-ResMulti-Trans	TSD-ResMulti-Trans	ProSelfLC-ResMulti-Trans	SelfRef-ResMulti-Trans	DKN
AWR	0.973	0.963	0.973	0.980	0.987	0.993	0.990	**0.993**	**0.993**
AF	0.400	0.333	0.400	0.400	0.400	0.400	0.400	0.400	**0.467**
BM	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**	**1.000**
CT	0.969	0.942	0.982	0.969	0.979	0.983	0.931	**0.998**	0.986
CK	0.903	0.889	0.944	0.944	0.944	0.944	0.944	0.944	**0.951**
DDG	0.320	0.420	0.420	0.500	0.520	**0.600**	0.520	0.540	0.560
EW	0.527	0.511	0.555	0.568	0.549	0.618	0.618	0.626	**0.628**
EP	0.920	0.883	0.971	0.964	**0.986**	0.978	0.978	0.979	0.979
EC	0.316	0.316	0.304	0.323	0.323	0.316	0.323	0.316	**0.372**
ER	0.881	0.874	0.915	0.919	0.930	0.919	0.919	0.930	**0.933**
FD	0.555	0.573	0.619	0.628	**0.640**	0.623	0.621	0.624	0.631
FM	0.580	0.570	**0.630**	0.620	0.580	0.630	0.540	0.590	0.600
HMD	0.541	0.508	0.419	0.481	0.500	0.500	0.481	0.500	**0.662**
HW	0.191	0.305	0.314	0.191	**0.357**	0.316	0.286	0.287	0.231
HB	0.619	0.658	0.737	0.727	0.717	0.727	0.727	0.727	**0.765**
IW	0.228	0.237	**0.430**	0.316	0.237	0.387	0.359	0.237	0.362
JV	0.916	0.768	0.926	0.928	0.928	0.924	0.926	0.926	**0.930**
LIB	0.833	0.817	0.850	0.870	0.894	0.870	0.894	0.870	**0.900**
LSST	0.391	0.265	0.440	0.456	**0.575**	0.551	0.161	0.265	0.347
MI	0.570	0.560	**0.640**	0.600	0.610	0.590	0.580	0.590	0.620
NATO	0.850	0.817	0.882	0.850	0.850	**0.883**	0.850	0.878	0.872
PD	0.892	0.892	0.975	0.939	0.930	0.930	0.939	0.939	**0.948**
PEMS	0.745	0.758	0.914	0.914	0.914	0.914	**0.930**	**0.930**	**0.930**
PS	0.288	0.269	0.304	0.318	0.418	0.418	0.304	0.439	**0.525**
RS	0.854	0.856	0.855	0.862	0.868	0.868	0.862	0.868	**0.879**
SRS1	0.867	0.862	0.884	0.899	0.899	0.908	0.908	0.899	**0.913**
SRS2	0.550	0.539	0.533	0.539	0.550	0.539	**0.600**	0.533	**0.600**
SAD	0.946	0.787	**0.979**	0.959	0.963	0.986	0.787	0.946	0.963
SWJ	0.400	0.400	**0.600**	0.533	0.533	**0.600**	0.533	0.533	0.533
UW	0.881	0.869	0.855	0.881	0.868	0.881	0.869	0.894	**0.897**
Win	0	0	5	0	4	2	0	1	**13**
Tie	1	1	1	1	1	2	3	3	**4**
Lose	29	29	24	29	25	26	27	26	**13**
Best	1	1	6	1	5	4	3	4	**17**
AVG_rank	7.150	8.050	5.050	4.933	4.267	3.767	5.167	4.267	**2.250**

datasets are shown in Table VII. First of all, let us compare DKN and ResMulti-Trans. In terms of 'win'/'tie'/'lose' and AVG_rank, DKN results in 13/4/13 and 2.250 while ResMulti-Trans obtains 5/1/24 and 5.050, which demonstrates the effectiveness of DDSD. To visualize the difference between DKN and ResMulti-Trans, we show the accuracy plot of DKN against ResMulti-Trans on the whole UEA2018 archive in Fig. 3. The results show that DKN obtains 'win'/'tie'/'loss' in 20/1/9 cases, respectively, illustrating that our DDSD well regularizes the DKN model and thus greatly improves its performance on MTSC.

Then, we compare DKN with those ResMulti-Trans models with other self-distillation techniques. It is no doubt that DDSD performs significantly better than BYOT, SAD, TSD, ProSelfLC, and SelfRef, in terms of 'win'/'tie'/'lose' and AVG_rank. The results, to a certain extent, demonstrate that our DDSD effectively promotes the mutual knowledge transfer between lower- and higher-level semantic information, helping DKN discover abundant representations and regularizations hidden in data.

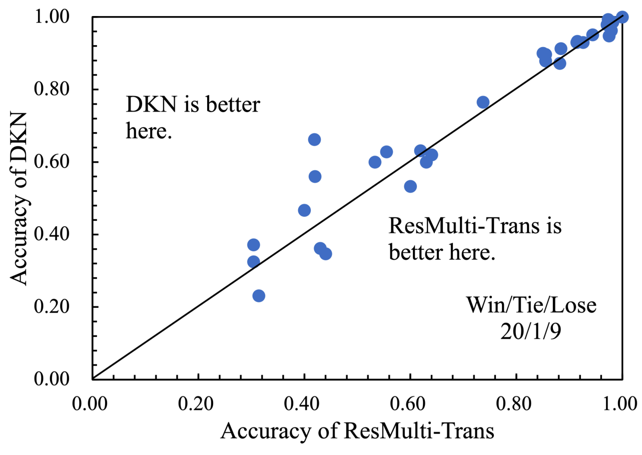


Fig. 3. Accuracy plot showing the performance difference between DKN and ResMulti-Trans on 30 UEA2018 datasets.

E. Experimental Analysis

To study the performance of DKN, we compare it with 18 existing MTSC algorithms:

- Three distance-based benchmark algorithms: ED_I, DTW_I, and DTW_D [7], [23].
- WM: the bag-of-pattern based approach with statistical feature selection, also called WEASEL+MUSE [40].
- CBOSS: the contractable bag of symbolic Fourier approximation symbols method [35].
- MLCN: the multivariate LSTM-fully convolutional network [14].
- RISE: the random interval spectral ensemble algorithm [26].

TABLE VIII
THE TOP-1 ACCURACY RESULTS OBTAINED BY VARIOUS MTSC ALGORITHMS ON 30 UEA2018 DATASETS.

Dataset Index	MLP	FCN	Inception Time	ResNet	ED_I	DTW_I	DTW_D	WM	CBOSS	MLCN	RISE	TSF	TapNet	XEM	CMFM + SVM	Mini ROCKET	DA-Net	Conv-GRU	DKN
AWR	0.043	0.823	0.897	0.943	0.970	0.980	0.987	**0.993**	0.990	0.957	0.963	0.953	0.987	**0.993**	0.973	0.992	0.980	0.973	**0.993**
AF	0.400	0.200	0.267	0.200	0.267	0.267	0.220	0.267	0.267	0.333	0.267	0.200	0.333	**0.467**	0.267	0.133	**0.467**	**0.467**	**0.467**
BM	0.875	**1.000**	**1.000**	**1.000**	0.676	**1.000**	0.975	**1.000**	**1.000**	0.875	**1.000**	**1.000**	**1.000**	**1.000**	0.975	**1.000**	0.925	**1.000**	**1.000**
CT	0.056	0.741	0.935	0.977	0.964	0.969	0.989	0.990	0.986	0.917	0.986	0.931	0.997	0.979	0.970	0.065	**0.998**	0.966	0.986
CK	0.111	0.917	0.958	0.958	0.944	0.986	**1.000**	0.986	N/A	N/A	N/A	N/A	0.958	0.986	0.958	0.986	0.861	0.943	0.951
DDG	0.360	0.600	0.560	0.600	0.275	0.550	0.600	0.575	0.480	0.380	0.220	0.460	0.575	0.375	0.420	**0.650**	0.520	0.540	0.560
EW	0.233	0.684	0.727	0.833	0.549	N/A	0.618	0.890	0.511	0.330	0.626	0.712	0.489	0.527	0.847	**0.954**	0.489	0.811	0.628
EP	0.312	0.935	0.935	0.964	0.666	0.978	0.964	0.993	0.979	0.732	0.979	**1.000**	0.971	0.986	0.978	**1.000**	0.883	0.978	0.979
EC	0.300	0.349	0.321	0.317	0.293	0.304	0.323	0.316	0.304	0.373	0.445	**0.487**	0.323	0.372	0.228	0.380	0.338	0.332	0.372
ER	0.159	0.778	0.822	0.907	0.133	0.133	0.133	0.133	0.919	0.941	0.881	0.859	0.133	0.200	0.930	**0.981**	0.874	0.400	0.933
FD	0.565	0.518	0.528	0.534	0.519	N/A	0.529	0.545	0.513	0.555	0.640	0.508	0.556	0.614	0.583	0.631	**0.648**	0.640	0.631
FM	0.510	0.510	0.500	0.460	0.550	0.520	0.530	0.540	0.519	0.580	0.581	0.562	0.530	0.590	0.460	0.450	0.510	0.580	**0.600**
HMD	0.216	0.270	0.392	0.216	0.278	0.306	0.231	0.378	0.292	0.544	0.481	0.312	0.378	0.649	0.284	0.392	0.365	0.338	**0.662**
HW	0.038	0.192	0.294	0.382	0.200	0.316	0.286	**0.531**	0.504	0.305	0.359	0.191	0.357	0.287	0.187	0.511	0.159	0.451	0.231
HB	0.665	0.724	0.717	0.716	0.619	0.658	0.717	0.727	0.564	0.458	0.535	0.518	0.751	0.761	0.727	**0.771**	0.624	0.746	0.765
IW	0.104	0.491	0.302	0.231	0.128	N/A	N/A	N/A	N/A	N/A	N/A	N/A	0.208	0.228	0.100	**0.595**	0.567	0.208	0.362
JV	0.114	0.941	0.949	0.924	0.924	0.959	0.949	0.978	N/A	N/A	N/A	N/A	0.965	0.978	0.778	0.989	0.938	**0.991**	0.930
LIB	0.078	0.478	0.744	0.844	0.833	0.894	0.870	0.894	0.894	0.850	0.806	0.806	0.850	0.772	0.817	0.878	0.800	0.889	**0.900**
LSST	0.326	0.337	0.290	0.232	0.456	0.575	0.551	0.628	0.458	0.390	0.161	0.265	0.568	**0.652**	**0.652**	0.643	0.560	0.548	0.347
MI	0.530	0.590	0.530	0.560	0.510	N/A	N/A	0.500	0.390	0.510	0.480	0.550	0.590	0.600	0.500	0.550	0.500	0.512	**0.620**
NATO	0.167	0.900	0.889	0.878	0.850	0.850	0.883	0.883	0.850	0.900	0.800	0.839	**0.939**	0.916	0.800	0.928	0.878	0.916	0.872
PD	0.211	0.970	0.977	0.973	0.973	0.939	0.977	0.969	0.939	0.979	0.892	0.831	**0.980**	0.977	0.665	0.965	**0.980**	0.939	0.948
PEMS	0.340	0.832	0.888	0.828	0.705	0.734	0.711	N/A	0.730	0.745	0.982	**0.994**	0.751	0.942	0.959	0.522	0.867	0.874	0.930
PS	0.414	0.466	0.466	0.466	0.104	0.151	0.151	0.190	0.151	0.151	0.137	0.269	0.175	0.288	0.247	0.292	0.093	0.215	**0.525**
RS	0.276	0.796	0.829	0.836	0.868	0.842	0.803	0.914	0.854	0.856	0.895	0.823	0.868	**0.941**	0.809	0.868	0.803	0.888	0.879
SRS1	0.686	0.805	0.805	0.761	0.771	0.765	0.775	0.744	0.765	0.908	0.840	0.724	0.652	0.839	0.771	0.874	**0.924**	0.843	0.913
SRS2	0.456	0.511	0.561	0.511	0.483	0.533	0.539	0.522	0.533	0.506	0.483	0.494	0.550	0.550	0.450	0.522	0.561	0.566	**0.600**
SAD	0.108	0.729	0.872	0.932	0.967	0.959	0.963	0.982	N/A	N/A	N/A	N/A	**0.983**	0.973	0.979	0.100	0.980	0.963	0.963
SWJ	0.200	0.267	0.133	0.133	0.200	0.333	0.200	0.333	0.333	0.400	0.333	0.267	0.400	0.400	0.267	0.333	0.400	0.426	**0.533**
UW	0.131	0.497	0.544	0.759	0.881	0.868	0.903	0.903	0.869	0.859	0.775	0.684	0.894	0.897	0.728	0.916	0.833	**0.919**	0.897
Win	0	0	0	0	0	0	1	1	0	0	0	2	2	1	0	5	3	2	**7**
Tie	0	1	1	1	0	1	1	2	1	0	1	2	2	**4**	1	1	2	2	3
Lose	30	29	29	29	30	29	28	27	29	30	29	26	26	25	29	24	25	26	**20**
Best	0	1	1	1	0	1	2	3	1	0	1	4	4	5	1	6	5	4	**10**
AVG_rank	15.483	11.350	10.267	10.783	13.050	11.317	10.567	7.800	11.583	11.050	11.017	12.567	7.567	5.883	11.433	6.400	9.467	6.867	**5.550**

- TSF: the time series forest algorithm for MTSC [32].
- MLP: the multilayer perceptron network for MTSC [42].
- FCN: the fully convolutional network for MTSC [42].
- InceptionTime: the Inception-based neural network for MTSC [41].
- ResNet: the residual neural network for MTSC [42].
- TapNet: the attentional prototype network integrating traditional and DL approaches [52].
- XEM: the explainable-by-design ensemble method with the boosting-bagging and bias-variance trade-off approaches [27].
- CMFM+SVM: the complexity measures and features method with an SVM classifier for MTSC [31].
- MiniROCKET: the very fast (almost) deterministic transform method [47].
- DA-Net: the dual attention-based network, consisting of the squeeze-excitation window attention layer and sparse self-attention within windows layer [12].
- Conv-GRU: the convolutional network with a gated linear units kernel [56].

The top-1 accuracy results obtained by various MTSC algorithms are shown in Table VIII. DKN performs the best, achieving a 'win'/'tie'/'loss' result of 7/3/20 and the smallest AVG_rank score, namely 5.550. There are mainly two reasons why DKN has remarkable performance. First, the DKN's feature extractor is dual-network-based. ResMulti is responsible for multi-scale local pattern extraction while Trans takes care of global pattern extraction. With ResMulti and Trans well designed, the feature extractor provides DKN with sufficient and well-diversified local and global features. Second, the DDSD technique strengthens the model's representation learning ability and regularizes DKN, by encouraging mutual knowledge transfer between lower- and higher-level semantic information. XEM is the second-best algorithm regarding AVG_rank. Its explicit boosting-bagging and bias-variance trade-off techniques help extract the inherent connections among the dimensions at different timestamps. MiniROCKET takes the second place among all compared algorithms according to 'best'. This is because MiniROCKET uses simple linear classifiers with random convolutional kernels to mine multi-scale representations from the input. On the other hand, MLP is obviously the worst benchmark algorithm against 'win'/'tie'/'loss' and AVG_rank since this multilayer-perceptron-based model often fails to extract as many promising representations from the data as possible. Besides, the AVG_rank results of various MTSC algorithms are shown in Fig. 4.

F. Computational Complexity

As suggested in [57], [58], we compare the proposed DKN with four single-network-based and two dual-network-based DL models regarding the number of parameters, floating point operations (FLOPs), and inference time on 30 UEA2018 testing datasets. These four single-network-based models are MLP [42], FCN [42], ResNet [42], and InceptionTime [41], while the two dual-network-based models include MLCN [14] and TapNet [52]. Table IX collects the testing results.

One can easily observe that DKN is slower than the four single-network-based models on most datasets, while the opposite situation appears on a few datasets, e.g., the inference time values of ResNet, InceptionTime, and DKN on

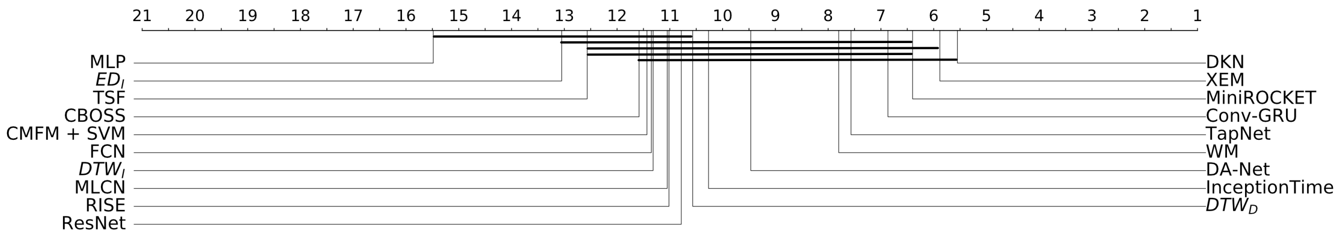


Fig. 4. AVG_rank results of various MTSC algorithms on 30 UEA2018 datasets.

TABLE IX
THE NUMBER OF PARAMETERS, FLOATING POINT OPERATIONS (FLOPS), AND INFERENCE TIME RESULTS WITH VARIOUS ALGORITHMS ON 30 UEA2018 TESTING DATASETS.

Metric	Method	AWR	AF	BM	CT	CK	DDG	EW	EP	EC	ER	FD	FM	HMD	HW	HB	IW	JV	LIB	LSST	MI	NATO	PD	PEMS	PS	RS	SRS1	SRS2	SAD	SWJ	UW
Parameter (M)	MLP	1.162	1.143	0.804	0.785	4.099	182.079	54.456	0.813	3.130	0.635	4.967	1.203	2.504	0.743	12.855	2.707	0.680	0.554	0.617	96.503	1.117	0.515	69.841	1.715	0.594	3.191	4.535	1.111	5.503	0.978
	FCN	**0.292**	**0.282**	**0.286**	**0.285**	**0.287**	**1.658**	**0.286**	**0.283**	**0.283**	**0.285**	**0.427**	**0.309**	**0.290**	**0.286**	**0.342**	**0.486**	**0.293**	**0.284**	**0.288**	**0.345**	**0.305**	**0.283**	**1.267**	**0.296**	**0.286**	**0.286**	**0.287**	**0.294**	**0.284**	**0.284**
	InceptionTime	1.596	1.589	1.591	1.593	1.592	2.062	1.591	1.590	1.590	1.590	1.639	1.598	1.592	1.594	1.610	1.660	1.594	1.591	1.593	1.611	1.597	1.590	1.928	1.599	1.591	1.590	1.591	1.594	1.590	1.590
	ResNet	1.240	1.229	1.234	1.233	1.235	2.777	1.234	1.231	1.231	1.232	1.393	1.259	1.239	1.233	1.297	1.458	1.242	1.231	1.235	1.301	1.255	1.230	2.337	1.244	1.234	1.234	1.235	1.243	1.232	1.231
	MLCN	0.386	0.367	0.331	0.326	0.754	25.934	7.199	0.327	0.623	0.306	1.111	0.421	0.556	0.322	1.972	0.923	0.327	0.294	0.309	12.684	0.404	0.288	10.883	0.463	0.304	0.636	0.810	0.384	0.929	0.349
	TapNet	2.926	3.143	2.876	2.928	3.446	5.706	12.033	2.924	3.715	2.856	3.137	2.893	3.038	2.919	3.142	3.239	2.857	2.851	2.853	4.477	2.890	2.827	4.861	2.982	2.840	3.281	3.414	2.893	4.099	2.984
	DKN	1.651	1.700	1.568	1.620	1.870	3.371	6.149	1.592	1.987	1.561	1.654	1.566	1.649	1.625	1.681	2.005	1.576	1.571	1.579	2.348	1.588	1.551	3.210	1.731	1.550	1.766	1.833	1.598	2.178	1.628
FLOPs (M)	MLP	2.321	2.283	1.604	8.194	1.566	364.155	108.909	1.622	6.257	1.266	9.930	2.402	5.004	1.482	25.707	5.410	1.357	1.105	1.230	193.002	2.230	1.026	139.679	3.426	1.184	6.378	9.066	2.219	11.003	1.953
	FCN	**0.588**	**0.568**	**0.576**	**0.578**	**0.574**	**3.319**	**0.576**	**0.570**	**0.570**	**0.572**	**0.858**	**0.621**	**0.584**	**0.575**	**0.688**	**0.975**	**0.590**	**0.571**	**0.579**	**0.694**	**0.613**	**0.569**	**2.537**	**0.595**	**0.576**	**0.575**	**0.578**	**0.592**	**0.572**	**0.571**
	InceptionTime	6.097	7.544	5.723	10.799	5.995	102.146	88.135	6.046	11.979	5.529	8.051	5.820	7.524	5.895	12.953	6.919	5.467	5.443	5.454	63.759	5.779	5.297	43.417	6.641	5.401	9.386	10.863	5.889	15.495	6.475
	ResNet	4.119	4.097	4.106	4.108	4.103	7.192	4.107	4.099	4.099	4.102	4.424	4.156	4.115	4.105	4.232	4.555	4.121	4.100	4.109	4.239	4.148	4.099	6.312	4.127	4.106	4.106	4.108	4.124	4.101	4.100
	MLCN	0.772	0.734	0.662	1.508	0.652	51.868	14.390	0.653	1.246	0.612	2.221	0.842	1.111	0.644	3.943	1.846	0.653	0.588	0.618	25.368	0.807	0.576	21.766	0.926	0.608	1.272	1.620	0.768	1.857	0.697
	TapNet	5.872	6.306	5.772	6.911	5.876	11.432	24.087	5.868	7.450	5.732	6.294	5.806	6.095	5.858	6.305	6.499	5.734	5.722	5.727	8.975	5.799	5.673	9.743	5.984	5.700	6.583	6.849	5.806	8.219	5.988
	DKN	3.230	3.386	3.120	3.703	3.180	6.723	12.279	3.168	3.959	3.102	3.298	3.122	3.283	3.174	3.352	3.978	3.122	3.096	3.114	4.685	3.155	3.071	6.397	3.353	3.084	3.522	3.655	3.164	4.343	3.230
With GPU (s)	MLP	**0.038**	**0.026**	**0.027**	**0.068**	**0.047**	**0.477**	**0.521**	**0.030**	**0.052**	**0.034**	**0.600**	**0.029**	**0.033**	**0.055**	**0.149**	2.235	**0.038**	**0.031**	**0.125**	**0.463**	**0.032**	**0.097**	**0.645**	**0.325**	**0.032**	**0.057**	**0.054**	**0.129**	**0.028**	**0.045**
	FCN	0.973	0.976	0.759	0.937	0.859	0.837	2.033	0.834	1.406	0.837	1.099	0.806	0.814	0.844	0.904	**1.355**	0.815	0.851	0.974	1.032	0.875	0.966	0.871	1.152	0.808	1.267	1.022	1.165	0.878	0.920
	InceptionTime	1.099	1.004	1.165	1.334	1.121	1.083	4.047	2.375	1.917	1.189	1.884	1.171	1.053	2.533	1.165	2.085	1.107	1.093	1.484	1.384	1.121	1.140	1.675	2.573	1.143	1.478	1.275	1.984	1.078	1.206
	ResNet	0.977	0.935	0.961	1.221	0.965	0.934	4.041	0.926	2.099	0.887	1.483	1.054	0.934	1.034	1.000	1.861	1.027	1.127	1.350	1.310	1.117	0.954	1.000	1.890	1.004	1.490	1.176	1.856	0.960	1.142
	MLCN	3.383	8.993	2.583	4.306	27.072	6.138	5.569	4.606	4.360	1.863	2.496	1.609	2.946	3.700	2.686	2.208	1.508	1.535	1.769	7.708	1.604	1.022	3.986	1.098	1.240	1.992	2.567	2.729	1.640	0.908
	TapNet	1.019	0.871	0.874	1.444	1.047	10.093	6.298	0.946	2.512	0.923	2.408	1.073	0.969	1.105	1.328	4.037	1.020	1.005	1.257	1.786	1.128	0.966	8.306	2.113	0.958	1.883	1.358	1.599	1.000	1.278
	DKN	1.284	1.159	1.292	1.406	1.245	2.131	3.398	2.435	1.777	1.306	2.721	1.302	1.155	2.601	1.309	10.324	1.226	1.248	1.696	1.592	1.245	1.306	3.058	3.128	1.252	1.494	1.349	1.975	1.193	1.293
With CPU (s)	MLP	**0.180**	0.170	0.181	**0.180**	**0.181**	**0.228**	**0.254**	**0.181**	**0.188**	**0.180**	**0.248**	**0.179**	**0.184**	**0.180**	**0.201**	**0.475**	0.182	0.172	**0.204**	**0.242**	0.177	**0.183**	**0.287**	**0.270**	0.181	**0.178**	**0.183**	**0.201**	**0.173**	**0.175**
	FCN	0.344	**0.113**	**0.079**	1.802	0.655	0.614	16.242	0.240	3.095	0.168	2.243	0.085	0.257	0.951	0.731	5.913	**0.128**	**0.102**	0.716	2.484	**0.113**	0.265	0.880	4.831	**0.081**	1.829	1.462	1.695	0.311	0.737
	InceptionTime	2.401	0.664	0.423	13.598	4.454	0.992	121.256	1.630	23.553	1.091	11.129	0.456	1.710	6.569	4.083	26.891	0.748	0.610	4.535	15.697	0.665	1.570	1.572	33.786	0.451	13.475	10.735	11.059	2.097	5.357
	ResNet	2.081	0.567	0.318	11.820	3.948	1.243	116.298	1.405	20.842	0.948	10.462	0.361	1.470	6.058	3.915	26.804	0.623	0.553	4.133	13.973	0.549	1.444	1.941	31.775	0.363	11.873	9.400	10.024	1.829	4.736
	MLCN	2.804	10.259	1.872	5.730	24.642	5.954	123.254	3.790	36.196	1.186	4.601	0.849	7.733	3.445	5.227	12.372	0.941	0.723	1.800	14.536	0.918	0.640	9.118	18.253	0.678	20.370	12.150	3.166	2.356	8.250
	TapNet	2.778	0.682	0.377	15.934	5.345	10.929	143.189	1.837	27.686	1.187	16.999	0.560	1.974	7.839	5.617	54.096	0.801	0.591	5.464	19.352	0.778	1.808	10.974	42.985	0.407	15.889	12.593	13.381	2.409	6.164
	DKN	1.980	0.726	0.577	9.958	3.470	12.249	85.752	1.400	17.034	1.056	21.503	0.611	1.465	4.982	3.341	85.656	0.804	0.682	3.457	11.533	0.744	1.401	22.076	24.887	0.569	9.949	7.950	8.224	1.767	4.307

GPU on the EigenWorms dataset are 4.041, 4.047, and 3.398, respectively. In addition, DKN is faster than MLCN and slower than TapNet on most datasets. Only on a few datasets does the opposite happen, e.g., the inference time values of MLCN, TapNet, and DKN on CPU on the Handwriting dataset are 3.700, 1.105, and 2.601, respectively.

G. Case Study

To better illustrate the concept of "knowledge-aware", we visualize the features in DKN on the Epilepsy dataset in Fig. 5. One can observe that lower-level semantic information, e.g., the output feature of Residual Multi-block 1 in Fig. 5 (b), has extensive feature resolution maps and rich position information, consisting of features such as contour, edge, color, texture, and shape. In contrast, higher-level semantic information, such as the output feature of ResMulti in Fig. 5 (d), is obtained from lower-level semantic information, which has a more significant perception view, richer combination information, rougher location information, and better discrimination ability. Higher-level semantic information can be well used in downstream tasks, such as classification. Within a representation hierarchy, the quality of the semantic information learned from lower and higher levels significantly affects a model's performance [15]. Almost all the existing models update their parameters by the BP method [16]. Lower-level semantic information is, to a certain extent, affected by higher-level semantic information. Thus, lower- and higher-level semantic information learns from and influences each other during the model learning process. The purpose of the "knowledge-aware" is to effectively promote the mutual learning between lower- and higher-level semantic information, helping the model mine rich regularizations and relationships hidden in the data.

V. CONCLUSION

The proposed DKN has two crucial components: the dual-network-based feature extractor and the dense dual self-distillation (DDSD). In the feature extractor, the ResMulti network can appropriately mine multi-scale local features, while the Trans network can reasonably identify the global relations among the features extracted. The proposed DDSD can regularize the model and improve its robustness by enabling mutual knowledge flow between lower- and higher-level semantic information. Trough an extensive experimental study, we observe that DDSD beats BYOT, SAD, TSD, ProSelfLC, and SelfRef, in terms of 'win'/'tie'/'lose' and AVG_rank. Besides, DKN wins in 10 out of the 30 UEA2018 datasets and obtains the smallest AVG_rank score, namely 5.550, among the 19 MTSC algorithms for comparison. The records reflect that our DKN has excellent potential when addressing various MTSC problems in the real world.

DKN has limitations. For example, unnecessary distillation computation in DDSD leads to additional resource consump-

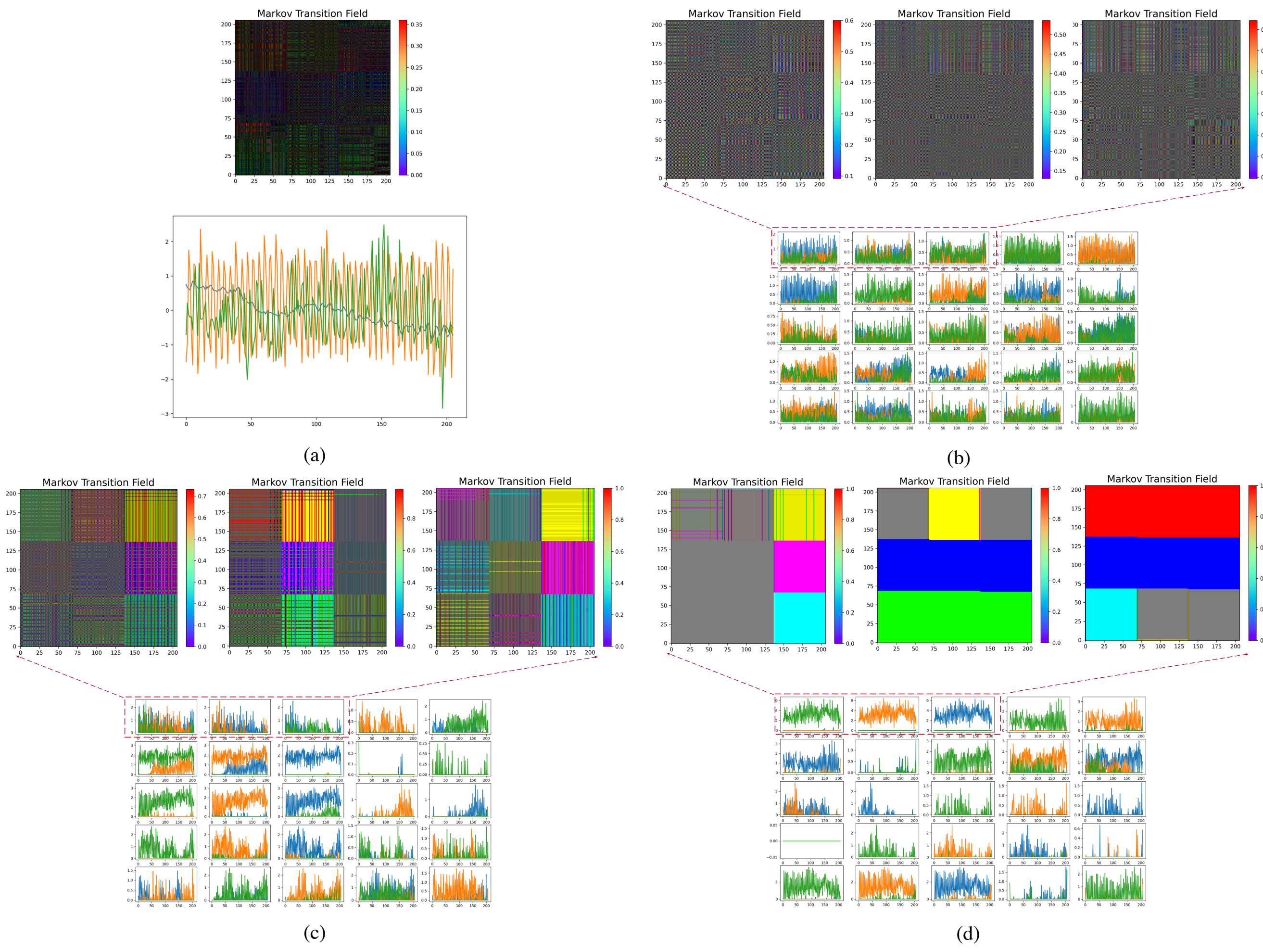


Fig. 5. Visualization of features in DKN on the Epilepsy dataset. (a) Visualization of the input sample; (b) visualization of the output feature of Residual Multi-block 1; (c) visualization of the output feature of Residual Multi-block 3; (d) visualization of the output feature of ResMulti.

tion. In the future, we will use an improved DDSD with a voting selection method to eliminate unnecessary distillation overhead. In the DKN's training, we adopt the fixed coefficient to integrate multiple loss functions, causing that parameters are locally optimized. To address this problem, we will apply multi-objective optimization to integrate multiple loss functions to gain near-optimal parameters.

Acknowledgment

The authors thank the editors and referees for their helpful suggestions on improving the quality of this paper.

References

[1] A. M. Roy, "Adaptive transfer learning-based multiscale feature fused deep convolutional neural network for eeg mi multiclassification in brain–computer interface," *Eng. Appl. Artif. Intel.*, vol. 116, pp. 1–17, 2022.

[2] Z. Gao, W. Dang, M. Liu, W. Guo, K. Ma, and G. Chen, "Classification of eeg signals on vep-based bci systems with broad learning," *IEEE Trans. Syst. Man Cy-S*, vol. 51, no. 11, pp. 7143–7151, 2021.

[3] K. Yan, Z. Ji, H. Lu, J. Huang, W. Shen, and Y. Xue, "Fast and accurate classification of time series data using extended elm: Application in fault diagnosis of air handling units," *IEEE Trans. Syst. Man Cy-S*, vol. 49, no. 7, pp. 1349–1356, 2019.

[4] B. Pourbabaee, M. J. Roshtkhari, and K. Khorasani, "Deep convolutional neural networks and learning ecg features for screening paroxysmal atrial fibrillation patients," *IEEE Trans. Syst. Man Cy-S*, vol. 48, no. 12, pp. 2095–2104, 2018.

[5] C. Yin, S. Zhang, J. Wang, and N. N. Xiong, "Anomaly detection based on convolutional recurrent autoencoder for iot time series," *IEEE Trans. Syst. Man Cy-S.*, vol. 52, no. 1, pp. 112–122, 2022.

[6] K. Denecke, S. Vaaheesan, and A. Arulnathan, "A mental health chatbot for regulating emotions (sermo) - concept and usability test," *IEEE Trans. Emerging Top. Com.*, vol. 9, no. 3, pp. 1170–1182, 2021.

[7] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Muller, "Deep learning for time series classification: a review," *Data Min. Knowl. Disc.*, vol. 33, pp. 917–963, 2019.

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

[9] H. Xing, Z. Xiao, R. Qu, Z. Zhu, and B. Zhao, "An efficient federated distillation learning system for multitask time series classification," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022.

[10] D. Lee, S. Lee, and H. Yu, "Learnable dynamic temporal pooling for time series classification," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 8288–8296.

[11] Q. Ma, S. Li, and G. W. Cottrell, "Adversarial joint-learning recurrent neural network for incomplete time series classification," *IEEE Trans. Pattern Anal.*, vol. 44, no. 4, pp. 1765–1776, 2022.

[12] R. Chen, X. Yan, S. Wang, and G. Xiao, "Da-net: Dual-attention network for multivariate time series classification," *Inf. Sci.*, vol. 610, pp. 472–487, 2022.

[13] Z. Xiao, X. Xu, H. Xing, S. Luo, P. Dai, and D. Zhan, "Rtfn: A robust

temporal feature network for time series classification," *Inf. Sci.*, vol. 571, pp. 65–86, 2021.

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

[15] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal.*, vol. 35, no. 8, pp. 1798–1828, 2013.

[16] P. Munro, "Backpropagation," *Sammut, C., Webb, G.I. (eds) Encyclopedia of Machine Learning*, 2011.

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

[18] L. Zhang, J. Song, A. Gao, J. Chen, C. Bao, and K. Ma, "Be your own teacher: Improve the performance of convolutional neural networks via self distillation," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2019, pp. 3712–3721.

[19] M. Ji, S. Shin, S. Hwang, G. Park, and I.-C. Moon, "Refine myself by teaching myself: Feature refinement via self-knowledge distillation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2021, pp. 10 659–10 668.

[20] Y. Hou, Z. Ma, C. Liu, and C. Change, "Learning lightweight lane detection cnns by self attention distillation," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2019, pp. 1013–1021.

[21] Z. Zhang, C. Bao, and K. Ma, "Self-distillation: Towards efficient and compact neural networks," *IEEE Trans. Pattern Anal.*, vol. 44, no. 8, pp. 4388–4403, 2022.

[22] X. Wang, Y. Hua, E. Kodirov, D. A. Clifton, and N. M. Robertson, "Proselflc: Progressive self label correction for training robust deep neural networks," in *Proc IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2021, pp. 752–761.

[23] 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.

[24] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Min. Knowl. Disc.*, vol. 29, pp. 565–592, 2015.

[25] 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.

[26] 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.

[27] 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.

[28] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "Hive-cote 2.0: a new meta ensemble for time series classifcation," *Mach. Learn.*, vol. 110, pp. 3211–3243, 2021.

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

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

[31] F. J. Baldán and J. M. Benítez, "Multivariate times series classification through an interpretable representation," *Inf. Sci.*, vol. 569, pp. 596–614, 2021.

[32] H. Deng, G. Runger, E. Tuv, and M. Vladimir, "A time series forest for classification and feature extraction," *Inf. Sci.*, vol. 239, pp. 142–153, 2013.

[33] 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.

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

[35] 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.

[36] G. He, X. Xin, R. Peng, M. Han, J. Wang, and X. Wu, "Online rule-based classifier learning on dynamic unlabeled multivariate time series data," *IEEE Trans. Syst. Man Cy-S.*, vol. 52, no. 2, pp. 1121–1134, 2022.

[37] G. He, B. Li, H. Wang, and W. Jiang, "Cost-effective active semi-supervised learning on multivariate time series data with crowds," *IEEE Trans. Syst. Man Cy-S.*, vol. 52, no. 3, pp. 1437–1450, 2022.

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

[39] 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.

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

[41] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. Schmidt, J. Weber, G. I. 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.

[42] Z. Wang, W. Yan, and T. Oates, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. Int. Jt. Conf. Neural Networks (IJCNN)*, 2017, pp. 1578–1585.

[43] 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.

[44] 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, pp. 1454–1495, 2020.

[45] G. Li, B. Choi, J. Xu *et al.*, "Shapenet: A shapelet-neural network approach for multivariate time series classification," in *Proc. AAAI Conf. Artif. Intell.*, 2021, pp. 8375–8383.

[46] 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.

[47] 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.

[48] 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.

[49] F. M. Bianchi, S. Scardapane, S. Løkse, and R. Jenssen, "Reservoir computing approaches for representation and classification of multivariate time series," *IEEE Trans. Neur. Net. Learn.*, vol. 32, no. 5, pp. 2169–2179, 2021.

[50] S. H. Huang, L. Xu, and C. Jiang, "Artificial intelligence and advanced time series classification: Residual attention net for cross-domain modeling," *Fintech with Artificial Intelligence, Big Data, and Blockchain, Blockchain Technologies*, 2021.

[51] 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.*, pp. 1–28, 2022.

[52] 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.

[53] Z. Xiao, X. Xu, H. Xing, R. Qu, F. Song, and B. Zhao, "Rnts: Robust neural temporal search for time series classification," in *Proc. Int. Jt. Conf. Neural Networks (IJCNN)*, 2021, pp. 1–8.

[54] 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.

[55] 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.

[56] C. Liu, J. Zhen, and W. Shan, "Time series classification based on convolutional network with a gated linear units kernel," *Eng. Appl. Artif. Intel.*, vol. 123, pp. 1–11, 2023.

[57] Z. Peng, W. Huang, S. Gu, L. Xie, Y. Wang, J. Jiao, and Q. Ye, "Conformer: Local features coupling global representations for visual recognition," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2021, pp. 357–366.

[58] X. Hu, L. Chu, J. Pei, W. Liu, and J. Bian, "Model complexity of deep learning: A survey," *Knowl. Inf. Syst.*, vol. 63, pp. 2585–2619, 2021.

Zhiwen Xiao (M) received the B.Eng. degree in network engineering from the Chengdu University of Information Technology, Chengdu, China, and the M.Eng. degree in computer science from the Northwest A & F University, 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.

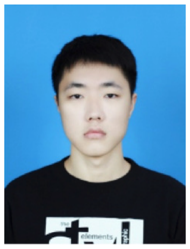
Huanlai Xing (M) received Ph.D. degree in computer science from University of Nottingham (Supervisor: Dr Rong Qu), Nottingham, U.K., in 2013. He was a Visiting Scholar in Computer Science, The University of Rhode Island (Supervisor: Dr. Haibo He), USA, in 2020-2021. Huanlai Xing is with the School of 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.

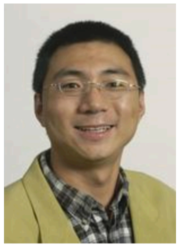
Rong Qu (SM'12) is a full Professor at the School of Computer Science, University of Nottingham. She received her Ph.D. in Computer Science from The University of Nottingham, U.K. in 2003. Dr. Qu is an associated editor at Engineering Applications of Artificial Intelligence, IEEE Computational Intelligence Magazine, IEEE Transactions on Evolutionary Computation, Journal of Operational Research Society and PeerJ Computer Science. She is the Vice-Chair of Evolutionary Computation Task Committee since 2019 and Technical Committee on Intelligent Systems Applications 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. Her research interests include the modelling and optimisation, personnel scheduling, network routing, machine learning, and data mining.

Li Feng received his PhD degree from Xi'an Jiaotong University 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.

Shouxi Luo (M) received the bachelor's degree in communication engineering and the Ph.D. degree in communication and information systems from the University of Electronic Science and Technology of China, Chengdu, China, in 2011 and 2016, respectively. He is an Associate Professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu. His research interests include data center networks, software-defined networking, and networked systems.

Penglin Dai (M) received the B.S. degree in mathematics and applied mathematics and the Ph.D. degree in computer science from Chongqing University, Chongqing, China, in 2012 and 2017, respectively. He is currently an Assistant Professor with the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China. His current research interests include intelligent transportation systems and vehicular cyber–physical systems.


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 PhD degree in the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. His research interests include semantic communication, federated learning (FL), deep reinforcement learning, cloud computing, and deep learning.


Yuanshun Dai (S'02–M'03) received the B.S. degree from Tsinghua University, Beijing, China, in 2000, and the Ph.D. degree from the National University of Singapore, Singapore, in 2003. He is currently the Dean of School of Computing and Artificial Intelligence, Southwest Jiaotong University, China. He was served as the Dean of School of Cybersecurity from 2018-2021, and School of Computer Science and Engineering from 2014-2018 at University of Electronic Science and Technology of China, where he was also a Chaired Professor and the Director of the National Key Laboratory in Next Generation Internet Technology. His current research interests include cloud computing, dependability, security, big data, and autonomic computing. He served as Chairs for 14 International Conferences, such as General Chairs for IEEE QRS, ISSSR, MASS, and Program Chair for IEEE PRDC'05, etc. He serves as an Associate Editor of the IEEE Transactions on Reliability, and is also on the editorial boards of several journals. He is continuously elected as "Chinese Most Cited Researchers" by Elsevier since 2015 till 2019 every year in the field of "Safety, Risk, Reliability and Quality".