Self-Bidirectional Decoupled Distillation for Time Series Classification

Zhiwen Xiao, *Member, IEEE*, Huanlai Xing, *Member, IEEE*, Rong Qu, *Senior Member, IEEE*, Hui Li, Li Feng, Bowen Zhao, and Jiayi Yang

***Abstract*—Over the years, many deep learning algorithms have been developed for time series classification (TSC). A learning model's performance usually depends on the quality of the semantic information extracted from lower and higher levels within the representation hierarchy. Efficiently promoting mutual learning between higher and lower levels is vital to enhance the model's performance during model learning. To this end, we propose a self-bidirectional decoupled distillation (Self-BiDecKD) method for TSC. Unlike most self-distillation algorithms that usually transfer the target-class knowledge from higher to lower levels, Self-BiDecKD encourages the output of the output layer and the output of each lower-level block to form a bidirectional decoupled knowledge distillation (KD) pair. The bidirectional decoupled KD promotes mutual learning between lower- and higher-level semantic information and extracts the knowledge hidden in the target and non-target classes, helping Self-BiDecKD capture rich representations from the data. Experimental results show that compared with a number of self-distillation algorithms, Self-BiDecKD wins 35 out of 85 UCR2018 datasets and achieves the smallest AVG_rank score, namely 3.2882. In particular, compared with a non-self-distillation Baseline, Self-BiDecKD results in 58/8/19 regarding 'win'/'tie'/'lose'.**

***Index Terms*—Convolutional Neural Network, Deep Learning, Data Mining, Knowledge Distillation, Time Series Classification**

IMPACT STATEMENT

Time series data has been widely applied to various real-world applications, e.g., sleep staging, electromyography signal classification, and arrhythmic heartbeat classification. The performance of a learning model relies on the representations captured from the data for download tasks, such as classification. In this paper, we introduce self-bidirectional decoupled distillation (Self-BiDecKD) for time series classification (TSC). Unlike most self-distillation algorithms that usually transfer the target-class knowledge from higher to lower levels, Self-BiDecKD encourages the output of the output layer and the output of each lower-level block to form a bidirectional decoupled knowledge distillation pair. When compared to several state-of-the-art self-distillation algorithms, Self-BiDecKD consistently demonstrates outstanding performance across a wide range of TSC applications.

Manuscript received 11, Sep., 2023; revised X, XX, XXXX; accepted 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, L. Feng, and B. Zhao 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, China (Emails: xiao1994zw@163.com; hxx@home.swjtu.edu.cn; fengli@swjtu.edu.cn; cn16bz@icloud.com).

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

H. Li is with the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an 710049, China (Email: lihui10@mail.xjtu.edu.cn).

J. Yang is with the School of Software Engineering, Tongji University 201804, China (Email: 2052522@tongji.edu.cn).

I. INTRODUCTION

TIME series data is a sequence of time-ordered data points associated with univariate or multiple time-dependent variables. It has been widely applied to areas, such as, sleep staging [1], electromyography and electroencephalogram signal analysis [2], arrhythmic heartbeat classification [3], and fault diagnosis [4]. It is crucial for an arbitrary time series classification (TSC) algorithm to extract both local and global patterns of data for various types of features [5], [6].

Over the years, deep learning algorithms for TSC have attracted extensive attention in the community because they can mine the intrinsic relationships among representations by constructing the internal representation hierarchy of data [7]. These algorithms can be roughly classified into single-network-based and dual-network-based models. A single-network-based model usually uses one (usually hybridized) network to extract the significant connections within the hierarchy, e.g., the residual network (ResNet) [8], multilayer perceptron (MLP) [8], fully convolution network (FCN) [8], ROCKET [9], ConvTimeNet [10], and InceptionTime [11]. In contrast, a dual-network-based model places one local-feature extraction network and one global-relation extraction network in parallel. The former, usually based on CNNs, focuses on the local features, while the latter captures the connections among the features already extracted, e.g., long short-term memory (LSTM)-FCN [12], SelfMatch [13], and robust neural temporal search network (RNTS) [14].

However, most deep learning models for TSC lack self reflection on their structures. A learning model's performance usually depends on quality of the semantic information extracted from lower and higher levels within the representation hierarchy [15]. It is known that higher-level semantic information is obtained from lower levels. Most learning models use the backpropagation (BP) approach to update their parameters [16]. According to BP, the parameter updates at lower levels are influenced by those at higher levels, i.e., higher-level semantic information can affect lower-level semantic information to a certain extent. *Therefore, lower- and higher-level*

semantic information learns from and influences each other during model learning. Efficiently promoting mutual learning between lower- and higher-level semantic information is crucial for improving the model's performance during model learning.

Recently, self-distillation has become popular in the knowledge distillation (KD) community. It itself acts as a student and a teacher at the same time, promoting knowledge flow within the model for the regularization purpose [17]. For example, BYOT transferred the knowledge extracted from the output layer to each lower-level block [18]. Ji *et al.* [19] proposed a feature refinement method that used self-knowledge distillation to transfer the refined knowledge for the classifier network through an auxiliary self-teacher network. Wang *et al.* [20] designed an end-to-end progressive self-label correction method, called ProSelfLC, to fit the noise and minimum entropy regularization principle. Zhang *et al.* [21] proposed two methods, namely transitive distillation and densely connected distillation, to enable the hierarchical knowledge transfer from higher to lower levels. However, most self-distillation algorithms suffer from two drawbacks listed below.

- They emphasize the knowledge transfer from higher to lower levels, ignoring the importance of lower-level semantic information to higher-level one, e.g., BYOT [18], ProSelfLC [20], transitive distillation [21], and densely connected distillation [21].
- They emphasize the target-class knowledge, ignoring the exploitation of "dark knowledge" hidden in the non-target class. However, such knowledge is also important to support efficient KD [23].

To overcome the two disadvantages above, we propose a self-bidirectional decoupled distillation (Self-BiDecKD) for TSC. Considering that lower- and higher-level semantic information learns from and influences each other during model learning, Self-BiDecKD pairs the output of the output layer with the output of each lower-level block to form a bidirectional decoupled KD structure.

Our significant contributions are summarized as follows.

- Unlike classical self-distillation algorithms that usually transfer the target-class knowledge from higher to lower levels, Self-BiDecKD's bidirectional decoupled KD enables mutual knowledge flow between lower and higher levels and also pays attention to the knowledge hidden in the target and non-target classes, helping Self-BiDecKD extract abundant representations from the data.
- Experimental results demonstrate that Self-BiDecKD outperforms six state-of-the-art self-distillation algorithms on 85 standard UCR2018 datasets regarding 'win'/'tie'/'lose' measure and AVG_rank, both based on the top-1 accuracy. Self-BiDecKD is a winner of 35 datasets, achieving the smallest AVG_rank score, namely 3.2882. In particular, compared with a non-self-distillation Baseline, Self-BiDecKD obtains 'win'/'tie'/'lose' in 58/8/19 cases, showing its potential in addressing various TSC problems.

The paper's remainder is listed below. Section II first reviews many classical TSC algorithms. Then, the key components of Self-BiDecKD are described in Section III. Section IV provides the experimental analysis and discussion. Finally, Section V summarizes a conclusion.

II. Related Work

This section reviews a number of existing traditional and deep learning-based TSC algorithms.

A. Traditional Algorithms

There are two mainstream algorithms in the TSC community, namely, Distance- and feature-based algorithms. Distance-based algorithms aim to capture the significant differences between different samples. The nearest neighbor (NN)- and dynamic time warping (DTW)-based algorithms are widely adopted distance-based algorithms [24], e.g., adaptive DTW (DTW_A), dependent DTW (DTW_D), and independent DTW (DTW_I). Many NN- and DTW-based ensemble algorithms have been an emerging trend for TSC. Many NN- and DTW-based ensemble algorithms have been an emerging trend for TSC. For instance, Lines *et al.* [25] introduced an elastic ensemble (EE) method integrating 11 classical classifier, e.g., weighted DTW, time warp with edit, and longest common subsequence, to solve various TSC problems. Bagnall *et al.* [26] proposed a transformation-based ensemble (COTE) method for feature extraction, fusing 37 classifiers constructed in the frequency, time, change, and shapelet transformation domains. Based on COTE, Lines *et al.* [27] designed a hierarchical structure with probabilistic voting to achieve decent performance on 85 benchmark datasets. Fauvel *et al.* [28] devised an explainable-by-design ensemble method based on boosting-bagging and divide-and-conquer approaches for multivariate TSC. Middlehurst *et al.* [29] presented an improved HIVE-COTE with temporal dictionary ensemble and Arsenal, namely HIVE-COTE 2.0, to extract the potential relationships and regularizations hidden in the data.

Feature-based algorithms aim to extract representative features from the input. For example, Pei *et al.* [30] proposed a time-series hidden-unit logistic method to capture temporal dependencies in the data. Tuncel *et al.* [31] designed an autoregressive forest ensemble algorithm to discover hierarchical relationships in the data. Baydogan and Runger [32] presented a local auto-pattern method to construct the dependency connection in time series. Large *et al.* [33] put forward a bag of symbolic Fourier approximation symbol method to model the spatial features in time series. Schäfer and Leser [34] devised a novel approach with word extraction and multivariate unsupervised symbols and derivatives for feature selection and weighting in time series. Wu *et al.* [35] put forward a broad fuzzy cognitive map system containing sparse autoencoder, high-order fuzzy cognitive map, and multilayer perceptron for TSC.

B. Deep Learning-based Algorithms

Deep learning algorithms aim to model a representation hierarchy of the given data, capturing the potential connections

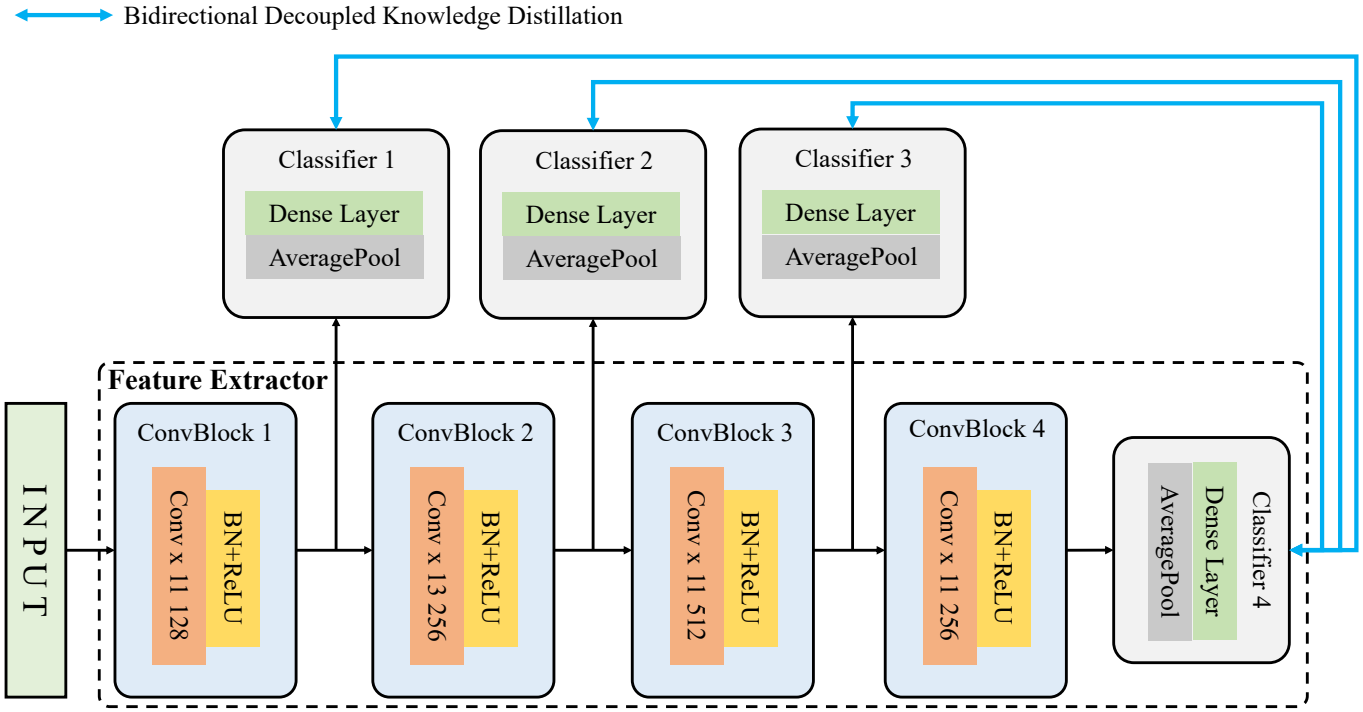


Fig. 1. The schematic diagram of Self-BiDecKD. Note that 'Conv x 11 128' represents a 1-dimensional convolutional neural network, where its filter and channel sizes are set to 11 and 128. 'BN' is a batch normalization module [22], and 'ReLU' is the rectified linear unit activation function.

among representations. Single- and dual-network-based models are two representative research methods. A single-network-based model uses one (often mixed) network to extract significant connections within a hierarchy. For example, Shi *et al.* [36] introduced a multi-relationship modeling module to capture the diversity of the relationships among classes. Chen *et al.* [37] devised a dual-attention-based network for local and global feature extraction. Xiao *et al.* [38] presented a multi-process collaborative architecture based on capsule networks to extract multi-scale features from the input. Li *et al.* [39] devised an embedding shapelet model based on deep neural networks called ShapeNet for discriminative shapelet selection. In [9], a random convolutional kernel classification approach (ROCKET) was used to address a series of TSC problems. Based on ROCKET, Dempster *et al.* [40] proposed an improved ROCKET (mini-ROCKET) for feature extraction. The typical single-based models include MLP [8], FCN [8], ResNet [8], InceptionTime [11], ConvTimeNet [10], adversarial joint-learning RNN [41], deep contrastive representation learning with self-distillation [42], and learnable dynamic temporal pooling [43]. On the contrary, a dual-network-based model usually contains two feature network in parallel, one for local-feature extraction and other for global-relation extraction. In [12], LSTM-FCN used its fully convolutional network and LSTM-based network for local-feature and global-relation extraction. In [44], a residual attention net contained a residual network and a transformer-based network to handle various TSC applications. In [45], a robust temporal neural network (RTFN) with a temporal feature network and an LSTM-based attention network achieved decent performance on a large number of multivariate and univariate time series datasets. The SelfMatch [13], RNTS [14], and attentional prototypical network [46], are representative dual-network-based models.

Many deep learning models for TSC overlook the interaction between lower and higher-level semantic information within the representation hierarchy. While most models emphasize learning higher-level semantic information from lower-level semantic information, they often underestimate the impact of lower-level semantic information on influencing the higher-level semantic information. In reality, both lower- and higher-level semantic information mutually influence each other during model learning. To end this, we introduce Self-BiDecKD, which facilitates mutual knowledge flow between lower and higher levels.

III. Self-BiDecKD

This section first defines the problem formulation, and it then describes the feature extraction and bidirectional decoupled KD components. Finally, the loss function is introduced.

A. Problem Formulation

Let $x_t = \{\{x_{1,1}^{(t)}, ..., x_{1,d}^{(t)}\}, ..., \{\{x_{l,1}^{(t)}, ..., x_{l,d}^{(t)}\}\} \in \mathcal{X}$ denote the t-th input sample, where $\mathcal{X} \subseteq \mathbb{R}^{l \times d}$ is the input space, and l and d denote the length and dimension of x_t, respectively. $y_t \in \mathcal{Y}$ is a categorical variable related to

x_t, where $\mathcal{Y}$ stands for the target space. This paper aims to train a predicted model $\mathcal{F} : \mathcal{X} \mapsto \mathcal{Y}$ on an arbitrary dataset, $\mathcal{D} = \{\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test}\}$. $\mathcal{D}_{train} = \{x_t, y_t\}_{t=1}^{n_{train}}$, $\mathcal{D}_{val} = \{x_t, y_t\}_{t=1}^{n_{val}}$, and $\mathcal{D}_{test} = \{x_t, y_t\}_{t=1}^{n_{test}}$ are the training, validation and testing data, respectively, where n_{train}, n_{val}, and n_{test} are the sizes of training, validation and testing data, respectively.

B. Feature Extractor

The feature extractor consists of four convolutional blocks (i.e., ConvBlock 1, ConvBlock 2, ConvBlock 3 and ConvBlock 4) and a classifier, as shown in Fig. 1. Each block contains a 1-dimensional CNN (Conv) module, a batch normalization (BN) module [22], and a rectified linear unit activation (ReLU) function, defined in equation. (1).

$$f_{ConvBlock}(x) = ReLU(BN(W_{cnn} \otimes x + b_{cnn})) \quad (1)$$

where W_{cnn} and b_{cnn} are the weight and bias vectors of CNN, respectively. $\otimes$ denotes the convolutional computation operation. $ReLU()$ and $BN()$ represent the ReLU and batch normalization functions, respectively.

The classifier is comprised of an average pooling layer and a dense (i.e., fully-connected) layer, mapping the features output by ConvBlock 4 to the corresponding labels.

C. Bidirectional Decoupled Knowledge Distillation

The bidirectional decoupled KD promotes mutual knowledge transfer between lower and higher levels and extracts the knowledge hidden in the target and the non-target classes, helping Self-BiDecKD mine rich representations from data.

Let $O_t^i = [O_{t,1}^i, O_{t,2}^i, ..., O_{t,C}^i] \in \mathbb{R}^{1\times C}$, $t = 1, 2, ..., n_{train}$, denote the t-th output vector of Classifier i, where C is the number of classes. Let $P_t^i = [P_{t,1}^i, P_{t,2}^i, ..., P_{t,C}^i] \in \mathbb{R}^{1\times C}$ and $P_{t,\setminus}^i = [P_{t,\setminus 1}^i, P_{t,\setminus 2}^i, ..., P_{t,\setminus C}^i] \in \mathbb{R}^{1\times C}$ be the classification probabilities of the target and non-target classes of O_t^i, respectively. $P_{t,j}^i \in P_t^i$ and $P_{t,\setminus j}^i \in P_{t,\setminus}^i$ $(j = 1, 2, ..., C)$ are defined as:

$$\begin{aligned} P_{t,j}^i &= \frac{exp(O_{t,j}^i/T)}{\sum_{k=1}^{C} exp(O_{t,k}^i/T)} \\ P_{t,\setminus j}^i &= \frac{\sum_{m=1,m\neq j}^{C} exp(O_{t,m}^i/T)}{\sum_{k=1}^{C} exp(O_{t,k}^i/T)} \end{aligned} \quad (2)$$

where T is a temperature scaling parameter. Following the literature [18], [19], [20], [21], we set T = 1.0 in this paper.

The loss of the bidirectional decoupled KD measures the difference between the classification probabilities of the target class of two given classifiers and the difference between the classification probabilities of the non-target class of the same classifiers above. Let $\mathcal{L}_{BTCL}^{i,t}$ and $\mathcal{L}_{BNCL}^{i,t}$, $i = 1, 2, 3$, be the bidirectional target class loss and the bidirectional non-target class loss of the t-th input sample between Classifier i and Classifier 4, x_t. Let $\mathcal{L}_{BDKD}^{i,t}$, $i = 1, 2, 3$, denote the bidirectional decoupled KD's loss of x_t. $\mathcal{L}_{BDKD}^{i,t}$ is a combination of $\mathcal{L}_{BTCL}^{i,t}$ and $\mathcal{L}_{BNCL}^{i,t}$ via the weighted sum method, defined in equation (3).

$$\begin{aligned} \mathcal{L}_{BDKD}^{i,t} &= \alpha\mathcal{L}_{BTCL}^{i,t} + \beta\mathcal{L}_{BNCL}^{i,t} \\ &= \alpha(KL(P_t^i, P_t^4) + KL(P_t^4, P_t^i)) \\ &+ \beta(KL(P_{t,\setminus}^i, P_{t,\setminus}^4) + KL(P_{t,\setminus}^4, P_{t,\setminus}^i)) \end{aligned} \quad (3)$$

where α and β are the weights of $\mathcal{L}_{BTCL}^{i,t}$ and $\mathcal{L}_{BNCL}^{i,t}$. $KL()$ represents the Kullback Leibler (KL) function. As the previous study suggested [23], we set α = 1.0. Meanwhile, based on our experimental results, we set β =1.0 (More details are found in Subsection IV-C).

D. Loss Function

The loss function of Self-BiDecKD, $\mathcal{L}$, comprises a supervised loss, $\mathcal{L}_{sup}$, and a bidirectional decoupled KD loss, $\mathcal{L}_{BDKD}$.

As the previous self-distillation work in [18], [19], [21], $\mathcal{L}_{sup}$ is the summation of the supervised loss functions on the four classifiers. The supervised loss on each classifier is based on the cross-entropy function, $CE()$, to measure the average difference between the ground truth labels and their prediction vectors. $\mathcal{L}_{sup}$ is defined in equation (4).

$$\mathcal{L}_{sup} = -\frac{1}{n_{train}} \sum_{i=1}^{4} \sum_{t=1}^{n_{train}} CE(P_t^i, y_t) \quad (4)$$

where y_t is a ground truth label associated with x_t.

As shown in Fig. 1, $\mathcal{L}_{BDKD}$ consists of the bidirectional decoupled KD loss between Classifier 1 and Classifier 4, the bidirectional decoupled KD loss between Classifier 2 and Classifier 4, and the bidirectional decoupled KD loss between Classifier 3 and Classifier 4, defined in equation (5).

$$\mathcal{L}_{BDKD} = -\frac{1}{n_{train}} \sum_{i=1}^{3} \sum_{t=1}^{n_{train}} \mathcal{L}_{BDKD}^{i,t} \quad (5)$$

The loss function of Self-BiDecKD, $\mathcal{L}$, is calculated as:

$$\begin{aligned} \mathcal{L} =& (1-\mu)\mathcal{L}_{sup} + \mu\mathcal{L}_{BDKD} + \lambda||\theta||_2^2 \\ =& -\frac{1}{n_{train}} \sum_{t=1}^{n_{train}} ((1-\mu)(\sum_{i=1}^{4} CE(P_t^i, y_t)) + \sum_{i=1}^{3} \mu\mathcal{L}_{BDKD}^{i,t}) \\ &+ \lambda||\theta||_2^2 \end{aligned} \quad (6)$$

where θ represents the parameters of Self-BiDecKD, μ $(0 < \mu < 1)$ is a coefficient of $\mathcal{L}$, and λ is a coefficient of $||\theta||_2^2$ (i.e., L_2 regularization). In the experiments, we set μ = 0.1 (More details are found in Subsection IV-C). Besides, the Self-BiDecKD's pseudo-code is given in Algorithm 1.

IV. Performance Evaluation

This section first explains the experimental setup and performance metrics. It then describes the hyper-parameter sensitivity, ablation study, and effects of different classifiers. Finally, the experimental results are analyzed.

TABLE I
DETAILS OF 85 UCR DATASETS.

ID	Type	Data Name	Test	Train	Length	Classes	ID	Type	Data Name	Test	Train	Length	Classes
1	Image	Adiac	391	390	176	37	44	Image	MedicalImages	760	381	99	10
2	Image	ArrowHead	175	36	251	3	45	Image	MiddlePhalanxOutlineAgeGroup	154	400	80	3
3	Spectro	Beef	30	30	470	5	46	Image	MiddlePhalanxOutlineCorrect	291	600	80	2
4	Image	BeetleFly	20	20	512	2	47	Image	MiddlePhalanxTW	154	399	80	6
5	Image	BirdChicken	20	20	512	2	48	Sensor	MoteStrain	1252	20	84	2
6	Sensor	Car	60	60	577	4	49	ECG	NonInvasiveFetalECGThorax1	1965	1800	750	42
7	Simulated	CBF	900	30	128	3	50	ECG	NonInvasiveFetalECGThorax2	1965	1800	750	42
8	Sensor	ChlorineConcentration	3840	467	166	3	51	Spectro	OliveOil	30	30	570	4
9	Sensor	CinCECGTorso	1380	40	1639	4	52	Image	OSULeaf	242	200	427	6
10	Spectro	Coffee	28	28	286	2	53	Image	PhalangesOutlinesCorrect	858	1800	80	2
11	Device	Computers	250	250	720	2	54	Sensor	Phoneme	1896	214	1024	39
12	Motion	CricketX	390	390	300	12	55	Sensor	Plane	105	105	144	7
13	Motion	CricketY	390	390	300	12	56	Image	ProximalPhalanxOutlineAgeGroup	205	400	80	3
14	Motion	CricketZ	390	390	300	12	57	Image	ProximalPhalanxOutlineCorrect	291	600	80	2
15	Image	DiatomSizeReduction	306	16	345	4	58	Image	ProximalPhalanxTW	205	400	80	6
16	Image	DistalPhalanxOutlineAgeGroup	139	400	80	3	59	Device	RefrigerationDevices	375	375	720	3
17	Image	DistalPhalanxOutlineCorrect	276	600	80	2	60	Device	ScreenType	375	375	720	3
18	Image	DistalPhalanxTW	139	400	80	6	61	Simulated	ShapeletSim	180	20	500	2
19	Sensor	Earthquakes	139	322	512	2	62	Image	ShapesAll	600	600	512	60
20	ECG	ECG200	100	100	96	2	63	Device	SmallKitchenAppliances	375	375	720	3
21	ECG	ECG5000	4500	500	140	5	64	Sensor	SonyAIBORobotSurface1	601	20	70	2
22	ECG	ECGFiveDays	861	23	136	2	65	Sensor	SonyAIBORobotSurface2	953	27	65	2
23	Device	ElectricDevices	7711	8926	96	7	66	Sensor	StarLightCurves	8236	1000	1024	3
24	Image	FaceAll	1690	560	131	14	67	Spectro	Strawberry	370	613	235	2
25	Image	FaceFour	88	24	350	4	68	Image	SwedishLeaf	625	500	128	15
26	Image	FacesUCR	2050	200	131	14	69	Image	Symbols	995	25	398	6
27	Image	FiftyWords	455	450	270	50	70	Simulated	SyntheticControl	300	300	60	6
28	Image	Fish	175	175	463	7	71	Motion	ToeSegmentation1	228	40	277	2
29	Sensor	FordA	1320	3601	500	2	72	Motion	ToeSegmentation2	130	36	343	2
30	Sensor	FordB	810	3636	500	2	73	Sensor	Trace	100	100	275	4
31	Motion	GunPoint	150	50	150	2	74	ECG	TwoLeadECG	1139	23	82	2
32	Spectro	Ham	105	109	431	2	75	Simulated	TwoPatterns	4000	1000	128	4
33	Image	HandOutlines	370	1000	2709	2	76	Motion	UWaveGestureLibraryAll	3582	896	945	8
34	Motion	Haptics	308	155	1092	5	77	Motion	UWaveGestureLibraryX	3582	896	315	8
35	Image	Herring	64	64	512	2	78	Motion	UWaveGestureLibraryY	3582	896	315	8
36	Motion	InlineSkate	550	100	1882	7	79	Motion	UWaveGestureLibraryZ	3582	896	315	8
37	Sensor	InsectWingbeatSound	1980	220	256	11	80	Sensor	Wafer	6164	1000	152	2
38	Sensor	ItalyPowerDemand	1029	67	24	2	81	Spectro	Wine	54	57	234	2
39	Device	LargeKitchenAppliances	375	375	720	3	82	Image	WordSynonyms	638	267	270	25
40	Sensor	Lightning2	61	60	637	2	83	Motion	Worms	77	181	900	5
41	Sensor	Lightning7	73	70	319	7	84	Motion	WormsTwoClass	77	181	900	2
42	Simulated	Mallat	2345	55	1024	8	85	Image	Yoga	3000	300	426	2
43	Spectro	Meat	60	60	448	3							

A. Experimental Setup

1) Data Description: Following the previous works [9], [11], [39], [40], we select 85 widely used datasets from the UCR 2018 archive [47]. The number of categories ranges from 2 to 60 and the length is from 24 to 2709, involving in multiple domains such as motion and electrocardiogram (ECG). More details about the datasets can be found in Table I.

2) Implementation Details: We set the decay value of BN to 0.9. To avoid overfitting during the training process, this paper adopts L_2 regularization. At the same time, we use the Adam optimizer with the momentum term, initial learning rate, and decay value set to 0.9, 0.001, and 0.9, respectively. All experiments are conducted on a workstation with Intel Xeon E5-2667V4 8 Core CPU $\times$ 2 3.2 GHz, 128 GB RAM, and 4 $\times$ Nvidia GTX Titan V 12G GPU.

B. Performance Metrics

To verify the Self-BiDecKD's performance, we use two widely used metrics: 'win'/'tie'/'lose' and AVG_rank, both based on the top-1 accuracy. As the previous works suggest [8], [9], [11], [12], [13], [14], [37], [39], [40], each algorithm's 'win', 'tie', and 'lose' values indicate it is better than, equivalent to, and worse than the other algorithms on how many datasets, respectively; The 'best' value is the summation of the corresponding 'win' and 'tie' values. Besides, following the studies in [8], [9], [11], [12], [13], [14], [40], we use the AVG_rank value to compare different algorithms regarding their performance on accuracy. The AVG_rank score is based on the Wilcoxon signed-rank test [48] with Holm's alpha (5%) correction.

C. Hyper-parameter Sensitivity

We use 85 UCR2018 datasets to study the impact of different hyper-parameter settings on the performance of Self-BiDecKD.

1) Self-BiDecKD with different β values: As aforementioned, β is the weight of the bidirectional non-target class loss. Table II shows the statistical results with different β (i.e., β = 0.1, 0.5, 1.0, 2.0, 5.0) values on 85 datasets. It is seen that

Algorithm 1 Self-BiDecKD

Input: $\mathcal{D} = (\mathcal{D}_{train}, \mathcal{D}_{val}, \mathcal{D}_{test})$;
Output: $\mathcal{Y}$;

```
Initialize the model parameters, θ_0;
//Training and validation
for k = 1 to n_ep do ▷ n_ep is the size of training epochs.
    Feedforward D_train into Self-BiDecKD;
    Obtain L_sup using equation (4);
    Obtain L_BDKD using equation (5);
    Obtain L using equation (6);
    Update θ_k using θ_k = θ_{k-1} − η∇_{θ_{k-1}} L(θ_{k-1}); ▷ η is the learning rate, and θ_{k-1} and ∇_{θ_{k-1}} denote the parameters and gradient at the (k-1)-th training epoch, respectively.
    if k > 1 then
        Validate Self-BiDecKD using D_val;
    end if
end for
// Testing the model
Use the trained model to predict Y of D_test.
```

TABLE II
THE STATISTICAL RESULTS WITH DIFFERENT β VALUES ON 85 DATASETS.

Metrics	β				
	0.1	0.5	1.0	2.0	5.0
Best	25	19	**38**	22	29
Win	13	12	**20**	6	14
Tie	12	7	**18**	16	15
Lose	60	66	**47**	63	56
AVG_rank	3.2471	2.9647	**2.4824**	2.8882	3.4176

β = 1.0 results in 20/18/47 regarding 'win'/'tie'/'lose' and the smallest AVG_rank, namely 2.4824. This means β = 1.0 is appropriate to help Self-BiDecKD mine rich representations from the data.

2) Self-BiDecKD with different μ values: μ is a coefficient of Self-BiDecKD to balance the supervised loss and the bidirectional decoupled KD loss. Table III shows the statistical results with different μ (i.e., μ = 0.1, 0.2, ..., 0.9) values on 85 datasets. One can observe that 0.1 is the best value for μ as it helps Self-BiDecKD to obtain the highest 'best' value, namely 49, and the smallest AVG_rank score, namely 2.8765.

3) Self-BiDecKD with different KD losses: An appropriate KD loss function is crucial for Self-BiDecKD to measure the knowledge difference between two given neural network layers. Table IV shows the statistical results with different losses on 85 datasets, including L_1 (Mean Absolute Error), L_2 (Mean Squared Error), CE (Cross Entropy), KL, JS (Jensen Shannon), and WD (Wasserstein Distance) losses. Clearly, KL outperforms the other 5 losses regarding 'best'/'win'/'tie'/'lose' and AVG_rank values. Therefore, we choose KL loss to promote the knowledge flow within the model.

D. Ablation Study

To study the effectiveness of the decoupled KD and bidirectional decoupled KD, we compare Self-BiDecKD with two of its variants listed below.

TABLE III
THE STATISTICAL RESULTS WITH DIFFERENT μ VALUES ON 85 DATASETS.

Metrics	μ								
	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9
Best	**49**	28	16	17	11	12	2	2	1
Win	**36**	14	5	6	3	3	0	0	0
Tie	13	**14**	11	11	8	9	2	2	2
Lose	**36**	57	69	68	74	73	83	83	84
AVG_rank	**2.8765**	3.1353	3.5118	3.7824	4.8235	5.5353	6.1529	7.0941	8.0882

TABLE IV
THE STATISTICAL RESULTS WITH DIFFERENT LOSSES ON 85 DATASETS.

Metrics	L_1	L_2	CE	KL	JS	WD
Best	19	20	28	**39**	31	26
Win	12	12	14	**20**	14	14
Tie	7	8	14	**19**	17	12
Lose	66	65	57	**46**	54	59
AVG_rank	4.8262	4.7782	3.3527	**2.3694**	2.8815	3.8834

- Self-UniDecKD: the unidirectional self-decoupled distillation method with the feature extractor in Fig. 1, which feeds back the knowledge obtained from the output layer to guide the learning process of each lower-level block via the decoupled KD loss function.
- Self-UniKD: the unidirectional self-distillation method with the feature extractor in Fig. 1, which feeds back the knowledge obtained from the output layer to guide the learning process of each lower-level block via the traditional self-distillation loss function, i.e., $KL()$.

1) Effectiveness of Decoupled Knowledge Distillation: To study the effectiveness of the decoupled KD, we compare Self-UniDecKD with Self-UniKD on 85 datasets. Fig. 2 depicts the accuracy plot of Self-UniDecKD against Self-UniKD. Compared with Self-UniKD that only considers the target-class knowledge, Self-UniDecKD pays attention to both the target- and non-target class knowledge. That is why Self-UniDecKD achieves 52/4/29 with respect to 'win'/'tie'/'lose' compared with Self-UniKD.

2) Effectiveness of Bidirectional Decoupled Knowledge Distillation: To study the effectiveness of the bidirectional decoupled KD, we compare Self-BiDecKD with Self-UniDecKD on 85 datasets. Fig. 3 depicts the accuracy plot of Self-BiDecKD against Self-UniDecKD. Compared with Self-UniDecKD, Self-BiDecKD obtains 47/8/30 regarding 'win'/'tie'/'lose'. That is because, unlike Self-UniDecKD that uses unidirectional decoupled KD, Self-BiDecKD applies bidirectional decoupled KD to promoting the mutual learning between lower- and higher-level semantic information, extracting abundant knowledge hiding in the data.

E. Impact of Different Classifiers

We also study the performance of different classifiers located at different network depths on 85 datasets. Fig. 4 shows the accuracy plot of two adjacent classifiers. One can observe that Classifier 2 achieves a 'win'/'tie'/'lose' of 79/2/4, compared with Classifier 1; Classifier 3 obtains a 'win'/'tie'/'lose' score of 82/3/0, compared with Classifier 2; Classifier 4 gets a

TABLE V
THE STATISTICAL RESULTS WITH VARIOUS SELF-DISTILLATION ALGORITHMS ON 85 DATASETS.

Metrics	Baseline	BYOT	TSD	SAD	ProSelfLC	SelfRef	ESD	Self-BiDecKD
Best	11	16	14	21	20	13	21	**35**
Win	4	6	4	10	5	6	10	**16**
Tie	7	10	10	11	15	7	11	**19**
Lose	74	69	71	64	65	72	64	**50**
AVG_rank	5.3882	4.5471	4.7589	4.5706	4.0471	4.7765	4.6235	**3.2882**

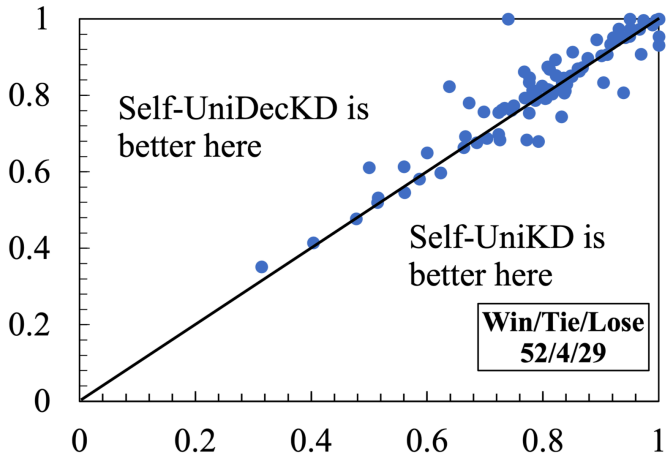


Fig. 2. Accuracy plot showing the performance difference between Self-UniDecKD and Self-UniKD on 85 datasets.

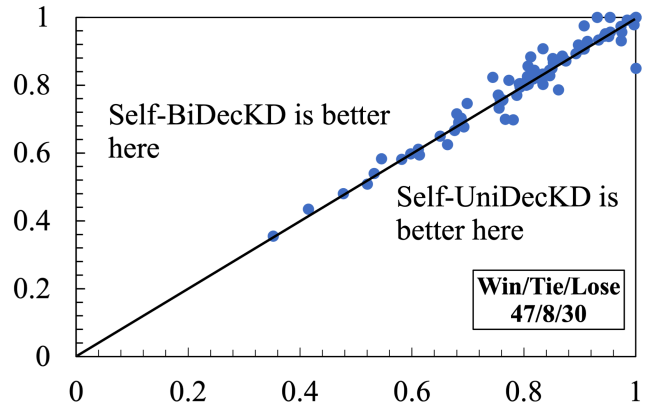


Fig. 3. Accuracy plot showing the performance difference between Self-BiDecKD and Self-UniDecKD on 85 datasets.

'win'/'tie'/'lose' score of 84/0/1, compared with Classifier 3. It means that a classifier closer to the output layer achieves better performance. That is because, as the network depth deepens, the features carried become richer and richer.

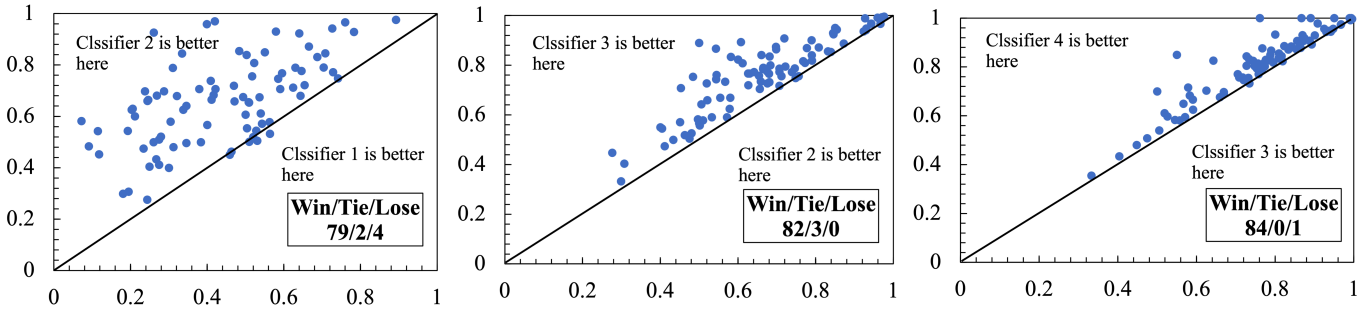
F. Result Analysis and Discussion

To evaluate the performance of Self-BiDecKD, we compare it with a number of state-of-the-art self-distillation algorithms below on the 85 UCR2018 datasets. Note that for fair comparison, the feature extractor used in each of these algorithms is the same as that in Fig. 1.

- Baseline: the proposed feature extractor in Fig. 1, which is a non-self-distillation model for TSC.
- BYOT: the best your own teacher [18] method adapted for TSC.
- TSD: the transitive self-distillation [21] method adapted for TSC.
- SAD: the layer-wise attention self-distillation [49] method adapted for TSC.
- ProSelfLC: the end-to-end progressive self-label correction [20] method adapted for TSC.
- SelfRef: the feature refinement self-distillation [19] method adapted for TSC.
- ESD: the ensemble self-distillation [21] method adapted for TSC.

Table V shows the statistical results with various self-distillation algorithms on 85 datasets. Self-BiDecKD is the best self-distillation algorithm among the eight algorithms as it obtains the best 'best'/'win'/'tie'/'lose' and the smallest AVG_rank values, namely, 35/16/19/50 and 3.2882. That is because that Self-BiDecKD uses the bidirectional decoupled KD to promote mutual knowledge transfer between lower and higher levels, efficiently learning the knowledge hidden in the target and non-target classes. ProSelfLC takes advantage of its progressive self-label correction to modify the target gradually and adaptively as training continues and to avoid entropy minimization. That brings it second in terms of 'win' and AVG_rank, namely 10 and 4.0471. On the contrary, it is difficult for SelfRef to extract sufficient features from the input through feature refinement only, resulting in its performing the worst among all the self-distillation algorithms for comparison.

To further study the effectiveness of Self-BiDecKD, we compare it with the non-self-distillation Baseline on 85 datasets. Fig. 5 depicts the accuracy plot of Self-BiDecKD against the Baseline. The results show that Self-BiDecKD achieves a 'win'/'tie'/'lose' of 58/8/19. That is because the decoupled KD efficiently promotes lower- and higher-level semantic information to learn from each other and helps Self-BiDecKD mine rich and diversified relationships and regularizations from the data. The AVG_rank results of the eight self-distillation algorithms are shown in Fig. 6.



(a) Classifier 2 vs. Classifier 1 (b) Classifier 3 vs. Classifier 2 (c) Classifier 4 vs. Classifier 3

Fig. 4. Accuracy plot results showing the performance difference between two adjacent classifiers on 85 datasets.

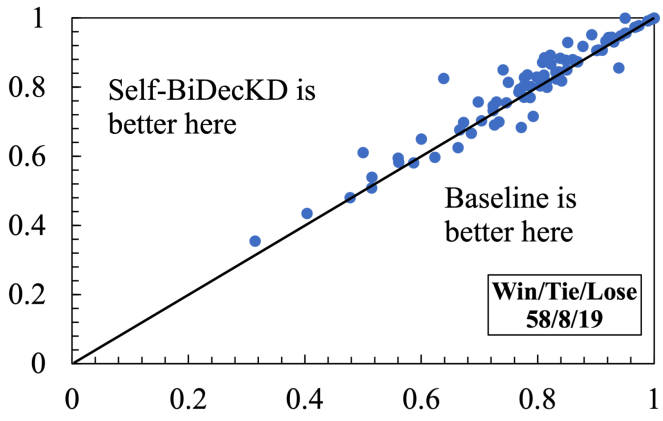


Fig. 5. Accuracy plot showing the performance difference between Self-BiDecKD and Baseline on 85 datasets.

G. Evaluation on ResNet and InceptionTime

To investigate the impact of Self-BiDecKD on different feature extractors, we consider two widely used feature extractors, i.e., ResNet [8] and InceptionTime [11].

- Baseline (ResNet): a residual convolutional network containing four residual blocks, which is a non-self-distillation model for TSC. Each residual block is comprised of three convolutional blocks, where their filter sizes are 8, 5, and 3, respectively.
- Self-BiDecKD (ResNet): the proposed self-bidirectional decoupled distillation, where ResNet is used as its feature extractor.
- Baseline (InceptionTime): an Inception convolutional network containing four Inception blocks, which is a non-self-distillation model for TSC. Each Inception block consists of 5 convolutional blocks and 1 maxpooling block, where the filter sizes of these convolutional blocks are 17, 11, 9, 5, and 3, respectively.
- Self-BiDecKD (InceptionTime): the proposed self-bidirectional decoupled distillation, where InceptionTime is used as its feature extractor.

To visualize the differences between Self-BiDecKD (ResNet) and Baseline (ResNet), we provide the accuracy plot of Self-BiDecKD (ResNet) against Baseline (ResNet) on 85 datasets in Fig. 7. The figure shows that Self-BiDecKD (ResNet) obtains 58/11/16 regarding 'win'/'tie'/'lose', reflecting that the proposed method promotes mutual knowledge flow between lower and higher levels within the model and is thus able to extract abundant representations from the data. Similarly, Self-BiDecKD (InceptionTime) beats Baseline (InceptionTime). As shown in Fig. 8, Self-BiDecKD (InceptionTime) obtains 57/10/18 in terms of 'win'/'tie/'lose', showing the effectiveness of Self-BiDecKD on knowledge transfer.

V. Conclusion

Self-BiDecKD enables the output of the output layer to form a bidirectional decoupled KD with the output of each lower-level block. While the knowledge captured by higher levels supervises lower levels, the bidirectional decoupled KD enables the knowledge extracted from lower levels to guide higher levels. Moreover, it pays attention to the knowledge hidden in the target and the non-target classes, which helps Self-BiDecKD to mine rich relationships and regularizations from the data. Experimental results show that compared with a number of self-distillation algorithms, Self-BiDecKD wins 35 out of 85 datasets and achieves the smallest AVG_rank score, namely 3.2882. In particular, compared with the non-self-distillation Baseline, Self-BiDecKD obtains a 'win'/'tie'/'lose' of 58/8/19, which unveils the potential of Self-BiDecKD to be applied to TSC problems in various real-world domains.

Acknowledgement

The authors extend their heartfelt gratitude to the editors and referees for their invaluable suggestions that have significantly enhanced the quality of this article.

References

[1] Z. Jia, X. Cai, G. Zheng, J. Wang, and Y. Lin, "Sleepprintnet: A multivariate multimodal neural network based on physiological time-series for automatic sleep staging," *IEEE Trans. Artif. Intell.*, vol. 1, no. 3, pp. 248–257, 2020.

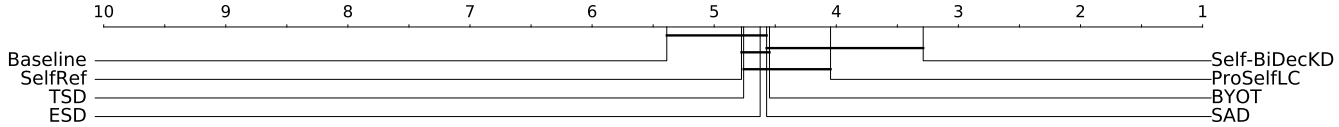


Fig. 6. AVG_rank results of various self-distillation algorithms on 85 datasets.

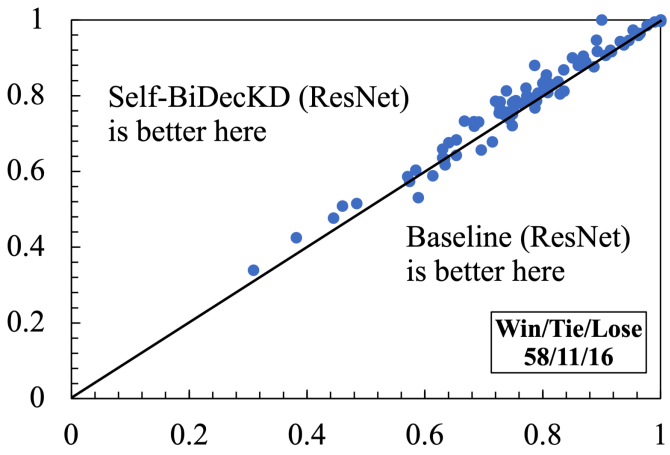


Fig. 7. Accuracy plot showing the performance difference between Self-BiDecKD (ResNet) and Baseline (ResNet) on 85 datasets.

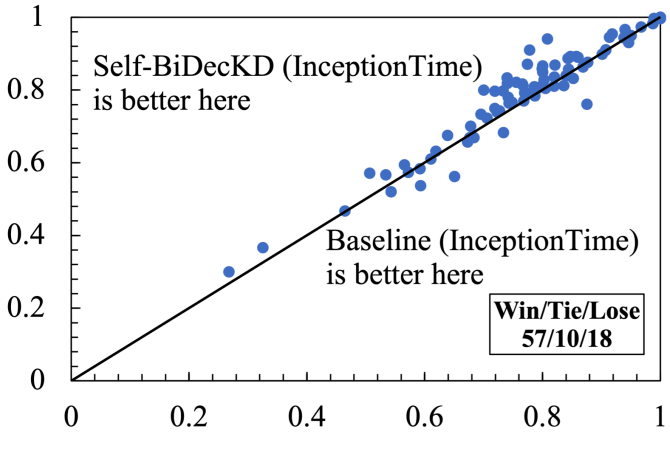


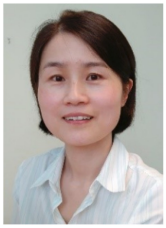
Fig. 8. Accuracy plot showing the performance difference between Self-BiDecKD (InceptionTime) and Baseline (InceptionTime) on 85 datasets.

[2] A. Erazo and S.-B. Ko, "A long short-term memory-based interconnected architecture for classification of grasp types using surface-electromyography signals," *IEEE Trans. Artif. Intell.*, pp. 1–14, 2023.
[3] S. Bhattacharyya, S. Majumder, P. Debnath, and M. Chanda, "Arrhythmic heartbeat classification using ensemble of random forest and support vector machine algorithm," *IEEE Trans. Artif. Intell.*, vol. 2, no. 3, pp. 260–268, 2021.
[4] G. Dewangan and S. Maurya, "Fault diagnosis of machines using deep convolutional beta-variational autoencoder," *IEEE Trans. Artif. Intell.*, vol. 3, no. 2, pp. 287–296, 2022.
[5] 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.
[6] H. Xing, Z. Xiao, 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.
[7] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, pp. 436–444, 2015.
[8] 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.
[9] 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.
[10] 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.
[11] H. I. Fawaz, B. Lucas, G. Forestier, C. Pelletier, D. F. 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.
[12] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate lstm-fcns for time series classification," *Neural Networks*, vol. 116, pp. 237–245, 2019.
[13] 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, pp. 8583–8610, 2022.
[14] 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.
[15] Y. Bengio, A. Courville, and P. Vincent, "Representation learning: A review and new perspectives," *IEEE Trans. Pattern Anal. Intell.*, vol. 35, no. 8, pp. 1798–1828, 2013.
[16] L. Jin, J. Cheng, J. Shi, and F. Huang, "Brief introduction of back propagation (bp) neural network algorithm and its improvement," *In: Jin, D., Lin, S. (eds) Advances in Computer Science and Information Engineering. Advances in Intelligent and Soft Computing*, 2012.
[17] J. Guo, B. Yu, S. Maybank, and D. Tao, "Knowledge distillation: A survey," *Int. J. Comput. Vision*, vol. 129, p. 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] 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.
[21] L. Zhang, C. Bao, and K. Ma, "Self-distillation: Towards efficient and compact neural networks," *IEEE Trans. Pattern Anal. Intell.*, vol. 44, no. 8, pp. 4388–4403, 2022.
[22] S. Ioffc and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," *arXiv preprint arXiv: 1502.03167*, 2015.
[23] B. Zhao, Q. Cui, R. Song, Y. Qiu, and J. Liang, "Decoupled knowledge distillation," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision Pattern Recognit. (CVPR)*, 2022, pp. 11 943–11 952.
[24] 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.

[25] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Min. Knowl. Disc.*, vol. 29, pp. 565–592, 2015.
[26] 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.
[27] 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.
[28] 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.
[29] M. Middlehurst, J. Large, M. Flynn, J. Lines, A. Bostrom, and A. Bagnall, "Hive-cote 2.0: a new meta ensemble for time series classification," *Mach. Learn.*, vol. 110, pp. 3211–3243, 2021.
[30] 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.
[31] K. S. Tuncel and M. G. Baydogan, "Autoregressive forests for multivariate time series modeling," *Pattern Recogn.*, vol. 73, pp. 202–215, 2018.
[32] 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.
[33] 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.
[34] P. Schäfer and U. Leser, "Multivariate time series classification with weasel+muse," *arXiv preprint arXiv:1711.11343*, 2017.
[35] 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.
[36] P. Shi, X. Dang, W. Ye, Z. Li, and Z. Qin, "Mrm2: Multi-relationship modeling module for multivariate time series classification," in *Proc. IEEE Int. Conf. Data Min. (ICDM)*, 2022, pp. 1185–1190.
[37] 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.
[38] Z. Xiao, X. Xu, H. Zhang, and E. Szczerbicki, "A new multi-process collaborative architecture for time series classification," *Knowledge-Based Syst.*, vol. 220, pp. 1–11, 2021.
[39] 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.
[40] 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.
[41] 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.
[42] 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.
[43] 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.
[44] 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.
[45] 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.
[46] 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.
[47] H. A. Dau, A. Bagnall, K. Kamgar, C.-C. M. Yeh, Y. Zhu, S. Gharghabi, C. A. Ratanamahatana, and E. Keogh, "The ucr time series archive," *IEEE/CAA J. Autom. Sin.*, vol. 6, no. 6, pp. 1293–1305, 2019.
[48] D. Rey and M. Neuhäuser, "Wilcoxon-signed-rank test," *In: Lovric, M. (eds) International Encyclopedia of Statistical Science*, 2021.
[49] Y. Hou, Z. Ma, C. Liu, and C. C. Loy, "Learning lightweight lane detection cnns by self attention distillation," in *Proc. IEEE Int. Conf. Comput. Vision (ICCV)*, 2019, pp. 1013–1021.

Zhiwen Xiao (Member, IEEE) 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 (Member, IEEE) 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.

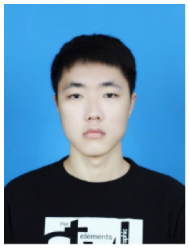


Rong Qu (Senior Member, IEEE) is a full Professor at the School of Computer Science, University of Nottingham. She received her B.Sc. in Computer Science and Its Applications from Xidian University, China in 1996 and Ph.D. in Computer Science from The University of Nottingham, U.K. in 2003. Her research 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 cyber security 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.

Hui Li received the B.Sc. and M.Sc. degrees in applied mathematics from the School of Mathematics and Statistics, Xi'an Jiaotong University, Xi'an, China, in 1999 and 2002, respectively, and the Ph.D. degree in computer science from the University of Essex, Colchester, U.K., in 2008. He is currently a Professor with the School of Mathematics and Statistics, Xi'an Jiaotong University. His current research interests include evolutionary computation, multiobjective optimization, and machine learning. Dr. Li was a recipient of the 2010 IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION Outstanding Paper Award as one of the inventors for MOEA/D.

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.



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 master's degree in the School of Information Science and Technology, Southwest Jiaotong University, Chengdu, China. His research interests include deep reinforcement learning, cloud computing, and deep learning.

Jiayi Yang is pursuing her bachelor's degree in software engineering at Tongji University, expecting to graduate in 2024. Her research interests include deep neural networks, reinforcement learning, and computer vision.