An Efficient Federated Distillation Learning System for Multitask Time Series Classification

Huanlai Xing, *Member, IEEE*, Zhiwen Xiao, *Member, IEEE*, Rong Qu, *Senior Member, IEEE*, Zonghai Zhu, and Bowen Zhao

***Abstract*—This article proposes an efficient federated distillation learning system (EFDLS) for multitask time series classification (TSC). EFDLS consists of a central server and multiple mobile users, where different users may run different TSC tasks. EFDLS has two novel components: a feature-based student–teacher (FBST) framework and a distance-based weights matching (DBWM) scheme. For each user, the FBST framework transfers knowledge from its teacher's hidden layers to its student's hidden layers via knowledge distillation, where the teacher and student have identical network structures. For each connected user, its student model's hidden layers' weights are uploaded to the EFDLS server periodically. The DBWM scheme is deployed on the server, with the least square distance (LSD) used to measure the similarity between the weights of two given models. This scheme finds a partner for each connected user such that the user's and its partner's weights are the closest among all the weights uploaded. The server exchanges and sends back the user's and its partner's weights to these two users which then load the received weights to their teachers' hidden layers. Experimental results show that compared with a number of state-of-the-art federated learning (FL) algorithms, our proposed EFDLS wins 20 out of 44 standard UCR2018 datasets and achieves the highest mean accuracy (70.14%) on these datasets. In particular, compared with a single-task baseline, EFDLS obtains 32/4/8 regarding "win"/"tie"/"lose" and results in an improvement of approximately 4% in terms of mean accuracy.**

***Index Terms*—Data mining, deep learning, federated learning (FL), knowledge distillation, time series classification (TSC).**

I. Introduction

Time series data is a series of time-ordered data points associated with one or more time-dependent variables and has been widely adopted in areas such as anomaly detection [1], [2], driving risk classification [3], service matching [4], electroencephalography (EEG) prediction [5], health care diagnosis [6], and emotion analysis [7]. A significant amount of research attention has been dedicated to time series classification (TSC) [8]. For example, Fawaz *et al.* [9] introduced a fully convolutional network (FCN) for local feature extraction. Zhang *et al.* [10] devised an attentional prototype network (TapNet) to capture rich representations from input data. Karim *et al.* [11] proposed a long short-term memory (LSTM) FCN (FCN-LSTM) for multivariate TSC. A robust temporal feature network (RTFN) hybridizing temporal feature network and LSTM-based attention network was applied to extract both the local and global patterns of data [12]. Li *et al.* [13] put forward a shapelet-neural network approach to mine highly diversified representative shapelets from the input. Lee *et al.* [14] designed a learnable dynamic temporal pooling method to reduce the temporal pooling size of the hidden representations obtained.

TSC algorithms are usually data-driven, where data come from various application domains [8], [9], [10], [11], [14]. Some data may contain private and sensitive information, such as bank accounts and ECG. However, traditional data collection operations could not protect such information, easily resulting in users' privacy leakage during the data collection and distribution processes involved in model training. To overcome the problem above, Google [15], [16], [17] invented federated learning (FL). FL allows users to collectively harvest the advantages of shared models trained from their local data without sending the original data to others. FederatedAveraging (FedAvg), federated transfer learning (FTL), and federated knowledge distillation (FKD) are the three mainstream research directions.

FedAvg calculates the average weights of the models of all users and shares the weights with each user in the FL system [18]. For instance, Ma *et al.* [19] devised a communication-efficient federated generalized tensor factorization for electronic health records. Liu *et al.* [20] used a federated adaptation framework to leverage the sparsity property of neural networks for generating privacy-preserving representations. A hierarchical personalized FL method aggregated heterogeneous user models, with considered privacy and model heterogeneity considered [21]. Yang *et al.* [22] modified the FedAvg method using partial networks for COVID-19 detection.

FTL introduces transfer learning techniques to promote knowledge transfer between different users, increasing system accuracy [23]. For example, Yang *et al.* [24] developed an FTL framework, FedSteg, for secure image steganalysis. An FTL method with dynamic gradient aggregation was proposed to weight the local gradients in the aggregation step

Manuscript received 19 April 2022; revised 5 August 2022; accepted 13 August 2022. Date of publication 24 August 2022; date of current version 7 September 2022. This work was supported in part by the Natural Science Foundation of Sichuan Province under Grant 2022NSFSC0568, in part by the National Natural Science Foundation of China under Grant 62172342, and in part by the Fundamental Research Funds for the Central Universities, China. The Associate Editor coordinating the review process was Qiang Miao. *(Corresponding author: Zhiwen Xiao.)*

Huanlai Xing, Zhiwen Xiao, Zonghai Zhu, and Bowen Zhao are with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu 611756, China (e-mail: hxx@home.swjtu.edu.cn; xiao1994zw@163.com; zzhu@swjtu.edu.cn; cn16bz@icloud.com).

Rong Qu is with the School of Computer Science, University of Nottingham, Nottingham NG7 2RD, U.K. (e-mail: rong.qu@nottingham.ac.uk).

Digital Object Identifier 10.1109/TIM.2022.3201203

for speech recognition [25]. Majeed *et al.* [26] proposed an FTL-based structure to address traffic classification problems.

Unlike FedAvg and FTL, FKD takes the average of all users' weights as the weights for all teachers and transfers each teacher's knowledge to its corresponding student via knowledge distillation (KD) [27]. A group knowledge transfer training algorithm was adopted to train small convolutional neural networks (CNNs) and transfer their knowledge to a prominent server-side CNN [28]. Mishra *et al.* [29] proposed a resource-aware FKD approach for network resource allocation. Itahara *et al.* [30] devised a distillation-based semi-supervised FL framework for communication-efficient collaborative training with private data. Nowadays, FKD is attracting increasingly more research attention.

In addition, there is a variety of FL-based algorithms in the literature. For instance, Chen *et al.* [31] applied asynchronous learning and temporally weighted aggregation to enhance system performance. Sattler *et al.* [32] presented a sparse ternary compression method to meet various requirements of the FL environment. A cooperative game based on a gradient algorithm was designed to tackle image classification and speech recognition tasks [33]. An ensemble FL system used a randomly selected subset of clients to learn multiple global models against malicious clients [34]. Hong *et al.* [35] combined adversarial learning and FL to produce federated adversarial debiasing for fair and transferable representations. Zhou *et al.* [36] proposed a privacy-preserving distributed contextual federated online learning framework with big data support for social recommender systems. Pan *et al.* [37] put forward a multigranular federated neural architecture search framework to automate the model architecture search in a federated and privacy-preserved manner.

Nowadays, most FL algorithms are developed for addressing single-task problems, where multiple users work together to complete one task, e.g., COVID-19 detection [22], traffic classification [26], or speech recognition [25]. It means the knowledge extracted from individual tasks is isolated. Actually, such knowledge has the potential to circulate in different task domains so as to benefit the classification performance of multiple tasks. For example, knowledge gained from the motion data of a smartwatch may effectively increase the accuracy of motion recognition on an environmental sensing instrument. However, sharing knowledge among different TSC tasks, i.e., the multitask TSC, has received little research attention in the literature. Unlike single-task TSC, multitask TSC aims to integrate different TSC tasks together into a framework, e.g., motion recognition on a smartwatch, gesture recognition on a gesture instrument, ECG detection on an ECG device, and driving behavior recognition on a driving instrument. In the framework, knowledge is shared by different TSC tasks running on different instruments to improve the accuracy of these tasks. Time series data is collected from various instruments, such as smartwatches, gesture instruments, and driving instruments. Each time series dataset has specific characteristics, e.g., length and variance, which may differ significantly from others. Thus, time series data is highly imbalanced and strongly non-independent, and identically distributed (non-I.I.D.). In multitask learning, it is commonly recognized that knowledge sharing among different tasks helps increase the efficiency and accuracy of each individual task [38]. *For most TSC algorithms, how to securely share knowledge of similar expertise among different tasks is still challenging. In other words, user privacy and knowledge sharing are two critical issues that need to be carefully addressed when devising practical multitask TSC algorithms.* To the best of our knowledge, FL for multitask TSC has received little research attention.

We present an efficient federated distillation learning system (EFDLS) for multitask TSC. This system consists of a central server and a number of mobile users running various TSC tasks simultaneously. Given two arbitrary users, they run either different tasks (e.g., ECG and motion) or the same task with different data sources to mimic real-world applications. EFDLS is characterized by a feature-based student–teacher (FBST) framework and a distance-based weights matching (DBWM) scheme. The FBST framework is deployed on each user, where the student and teacher models have identical network structures. For each user, its teacher's hidden layers' knowledge is transferred to its student's hidden layers, helping the student mine high-quality features from the data. The DBWM scheme is deployed on the EFDLS server, where the least square distance (LSD) is used to measure the similarity between the weights of two models. When all connected users' weights are uploaded completely, for an arbitrary connected user, the DBWM scheme finds the one with the most similar weights among all connected users. After that, the server sends the connected user's weights to the found one, which loads the weights to its teacher model's hidden layers.

Our main contributions are summarized below.

1) We propose EFDLS for multitask TSC, where each user runs one TSC task and different users may run various TSC tasks. The data generated on different users is different. In EFDLS, feature-based knowledge distillation is used for knowledge transfer in each user. Unlike the traditional FKD that adopts the average weights of all users to supervise the feature extraction process in each user, EFDLS finds the one with the most similar expertise (i.e., a partner) for each user according to LSD and offers knowledge sharing between the user and its partner. EFDLS aims at providing secure knowledge sharing of similar expertise among different tasks. To our best, this problem has not attracted enough research attention.
2) Experimental results demonstrate that EFDLS outperforms six state-of-the-art FL algorithms regarding the mean accuracy, "win"/"tie"/"lose" measure, and AVG_rank, which are all based on the top-1 accuracy, where 44 well-known UCR2018 datasets are considered. To be specific, EFDLS wins 20 out of 44 datasets and achieves the highest mean accuracy, namely 70.14%, on these datasets. Besides, compared with a single-task baseline, EFDLS obtains 32/4/8 regarding "win"/"tie"/"lose" and results in an improvement of approximately 4% in terms of mean accuracy. The results show the effectiveness of EFDLS in addressing TSC problems.

The rest of the article is organized below. Section II reviews the existing TSC algorithms. Section III overviews the architecture of EFDLS and describes its key components. Section IV provides and analyzes the experimental results, and conclusion is drawn in Section V.

II. Related Work

A large number of traditional and deep learning algorithms have been developed for TSC.

A. Traditional Algorithms

Two representative streams of algorithms are distance- and feature-based. For distance-based algorithms, it is quite common to combine the dynamic time warping (DTW) and nearest neighbor (NN), e.g., DTW_A, DTW_I, and DTW_D [39]. Besides, a significant number of DTW-NN-based ensemble algorithms have been proposed in the TSC community. For example, Lines and Bagnall [40] presented an elastic ensemble (EE) algorithm for feature extraction, with 11 types of 1-NN-based elastic distance considered. A collective of the transformation-based ensemble (COTE) with 37 NN-based classifiers was adopted to address various TSC problems [41]. The hierarchical vote collective of transformation-based ensembles (HIVE-COTE) [42] and local cascade ensemble [43] are two representative ensemble algorithms in the literature.

For feature-based algorithms, their aim is to capture sufficient discriminate features from the given data. For instance, Lines *et al.* [44] introduced a shapelet transformation method to find representative shapelets that reflected the trend of raw data. A bag-of-features representation framework was used to extract the information at different locations of sequences [45]. Dempster *et al.* [46] applied minimally random convolutional kernel transform to exploring the transformed features from data. In addition, the learned pattern similarity [47], bag of symbolic Fourier approximation symbols [48], hidden-unit logistic model [49], time series forest [50], and multifeature dictionary representation and ensemble learning [51] are also well-known algorithms.

B. Deep Learning Algorithms

By unfolding the internal representation hierarchy of data, deep learning algorithms focus on extracting the intrinsic connections among representations. Most of the existing deep learning models are either single-network- or dual-network-based [12]. A single-network-based model captures the significant correlations within the representation hierarchy of data by one (usually hybridized) network, e.g., FCN [9], ResNet [9], shapelet-neural network [13], InceptionTime [52], dynamic temporal pooling [14], multiprocess collaborative architecture [53], and multiscale attention convolutional neural network [54]. In contrast, a dual-network-based model usually consists of two parallel networks, i.e., local-feature extraction network (LFN) and global-relation extraction network (GRN), such as FCN-LSTM [11], RTFN [12], ResNet-Transformer [55], RNTS [56], SelfMatch [57], and TapNet [10].

Almost all algorithms above emphasized single-task TSC, e.g., traffic or gesture classification. However, TSC usually involves multiple tasks in real-world scenarios, like various applications with different TSC tasks run on different mobile devices in a mobile computing environment. Enabling efficient knowledge sharing of similar expertise among different tasks helps increase the average accuracy of these tasks. Nevertheless, sharing knowledge among different TSC tasks securely and efficiently is still a challenge. That is what FL aims for.

III. Efficient Federated Distillation Learning System

This section first overviews the architecture of EFDLS. Then, it introduces the FBST framework, DBWM scheme, and communication overhead.

A. System Overview

EFDLS is a secure distributed system for multitask TSC. There is a central server and multiple mobile users. Let N_{tot} and N_{conn} denote the numbers of total and connected users in the system, respectively, where $N_{\text{conn}} \leq N_{\text{tot}}$. Each user runs one TSC task at a time and different users might run different TSC tasks. For two arbitrary users, they run two different tasks, such as gesture and ECG classification, or the same task with different data sources.

The overview of EFDLS is shown in Fig. 1. In the system, users train their models locally based on knowledge distillation and share their model weights with users with similar expertise via the server. We propose FBST, a feature-based student–teacher framework that is deployed on each user as its learning model. For each user, its teacher's hidden layers' knowledge is transferred to its student's hidden layers. For each connected user, its student model's hidden layers' weights are uploaded to the EFDLS server periodically. We propose DBWM, a distance-based weights matching scheme deployed on the server, with the LSD adopted to measure the similarity between the weights of two given models. After the weights of all connected users are uploaded completely, for each connected user, the DBWM scheme is launched to find the one with the most similar weights among all connected users. In this way, every user has a partner to match with. For each connected user, its uploaded weights are sent to its partner that then loads these weights to its teacher model's hidden layers. The server's role looks like a telecom network switch. The EFDLS system allows users to benefit from knowledge sharing without sacrificing security and privacy.

B. Feature-Based Student–Teacher Framework

In the FBST framework, the student and teacher models have identical network structures. In each user, feature-based KD promotes knowledge transfer from the teacher's hidden layers to its student's hidden layers, helping the student capture rich and valuable representations from the input data.

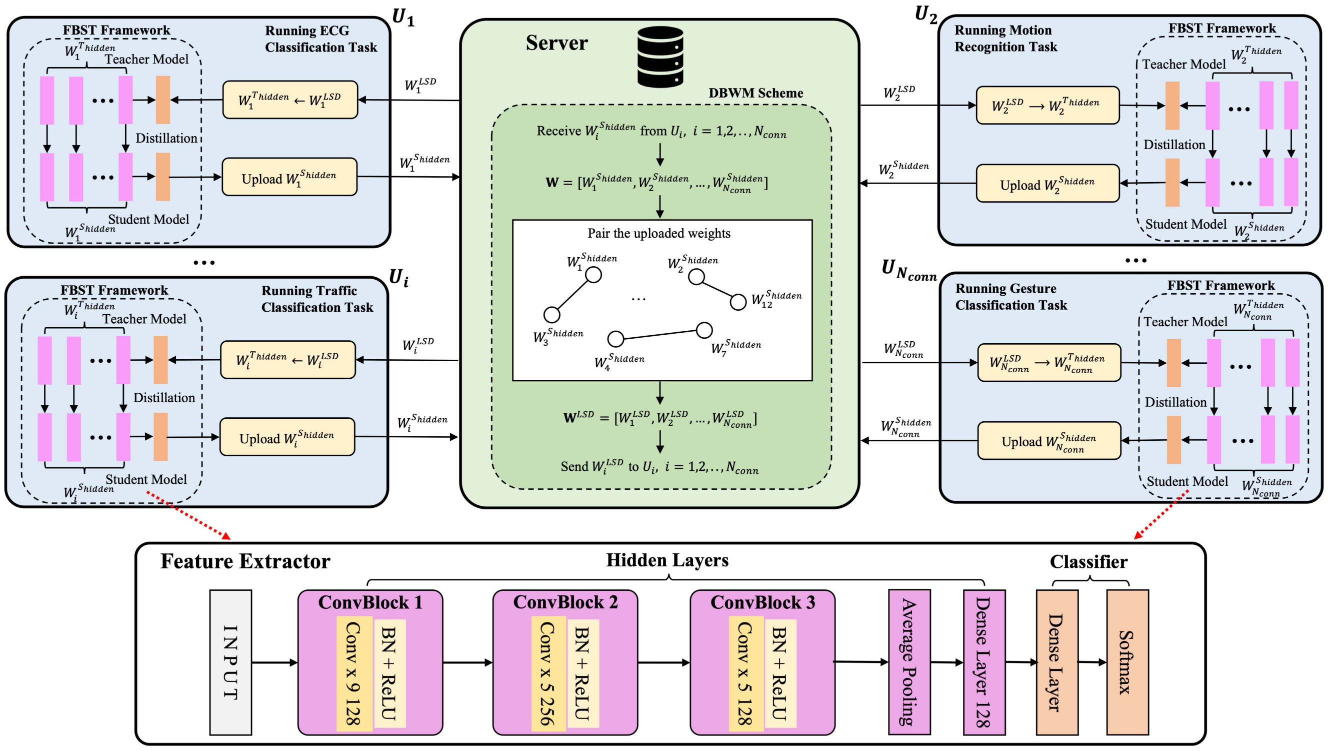


Fig. 1. Schematic of EFDLS. Note that “FBST Framework” and “DBWM Scheme” denote the FBST framework deployed on each user and the DBWM scheme run on the server. “Conv x 9 128” represents a 1-D convolutional neural network, where its filter and channel sizes are set to 9 and 128. “BN” is the batch normalization module, and “ReLU” is the rectified linear unit activation function.

1) Feature Extractor: The feature extractor contains multiple hidden layers and a classifier, as shown in Fig. 1. The hidden layers are responsible for local-feature extraction, including three convolutional blocks (i.e., ConvBlock1, ConvBlock2, and ConvBlock3), an average pooling layer, and a dense (i.e., fully connected) layer. Each ConvBlock consists of a 1-D CNN (Conv) module, a batch normalization (BN) module, and a rectified linear unit activation (ReLU) function, defined as

$$f_{\text{convblock}}(x) = f_{\text{relu}}(f_{bn}(W_{\text{conv}} \otimes x + b_{\text{conv}})) \quad (1)$$

where, W_{conv} and b_{conv} are the weight and bias matrices of CNN, respectively. $\otimes$ represents the convolutional computation operation. f_{bn} and f_{relu} denote the BN and ReLU functions, respectively.

Let $x_{bn} = \{x_1, x_2, \ldots, x_{N_{bn}}\}$ denote the input of BN, where x_i and N_{bn} stand for the ith instance and batch size, respectively. $f_{bn}(x_{bn})$ is defined in the following:

$$\begin{aligned} f_{bn}(x_{bn}) &= f_{bn}\left(x_1, x_2, \ldots, x_{N_{bn}}\right) \\ &= \left(\alpha \frac{x_1 - \mu}{\delta + \zeta} + \beta, \alpha \frac{x_2 - \mu}{\delta + \zeta} \right. \\ &\qquad \left. + \beta, \ldots, \alpha \frac{x_{N_{bn}} - \mu}{\delta + \zeta} + \beta \right) \\ \mu &= \frac{1}{N_{bn}} \sum_{i=1}^{N_{bn}} x_i \\ \delta &= \sqrt{\sum_{i=1}^{N_{bn}} (x_i - \mu)^2} \end{aligned} \quad (2)$$

where, μ and δ represent the mean and standard deviation of x_{bn}, respectively. $\alpha \in \mathbb{R}^+$ and $\beta \in \mathbb{R}$ are the parameters to be learned during training. $\zeta > 0$ is an arbitrarily small number.

The classifier is composed of a dense layer and a Softmax function, mapping high-level features extracted from the hidden layers to the corresponding labels.

2) Knowledge Distillation: Feature-based KD regularizes a student model by transferring knowledge from the corresponding teacher's hidden layers to the student's hidden layers [58]. For an arbitrary user, its student model captures sufficient discriminate representations from the data under its teacher model's supervision.

Let $O_i^{T,1}$, $O_i^{T,2}$, $O_i^{T,3}$, and $O_i^{T,4}$ be the outputs of ConvBlock 1, ConvBlock 2, ConvBlock 3, and the dense layer of the teacher's hidden layers. Let $O_i^{S,1}$, $O_i^{S,2}$, $O_i^{S,3}$, and $O_i^{S,4}$ be the outputs of ConvBlock 1, ConvBlock 2, ConvBlock 3, and the dense layer of the student's hidden layers. Following the previous work [28], we define the KD loss, $\mathcal{L}_i^{KD}$, of U_i as:

$$\mathcal{L}_i^{KD} = \sum_{m=1}^{4} \left\| O_i^{T,m} - O_i^{S,m} \right\|^2. \quad (3)$$

For U_i, its total loss, $\mathcal{L}_i$, consists of KD loss, $\mathcal{L}_i^{KD}$, and supervised loss, $\mathcal{L}_i^{\text{Sup}}$. As suggested in [10], [11], and [12], $\mathcal{L}_i^{\text{Sup}}$ uses the cross-entropy function to measure the average difference between the ground-truth labels and their prediction vectors, as shown in the following:

$$\mathcal{L}_i^{\text{Sup}} = -\frac{1}{N_{\text{seg}}} \sum_{j=1}^{N_{\text{seg}}} y_j \log(\hat{y}_j) \quad (4)$$

where, N_{seg} is the number of input vectors, and y_i and $\hat{y}_j$ are the ground-truth label and prediction vector of the jth input vector, respectively.

Inspired by the loss function of most KD algorithms [58], [59], we define the total loss of U_i, $\mathcal{L}_i$, as

$$\mathcal{L}_i = \epsilon \times \mathcal{L}_i^{\text{Sup}} + (1 - \epsilon) \times \mathcal{L}_i^{KD} \tag{5}$$

where, $\epsilon \in (0, 1)$ is a coefficient to balance $\mathcal{L}_i^{\text{Sup}}$ and $\mathcal{L}_i^{KD}$. In this article, we set $\epsilon = 0.9$ (more details can be found in Section IV-C).

C. Distance-Based Weights Matching

Identical to the NN and DTW, the LSD calculates the similarity between the weights of two given models using the Euclidean distance. When the weights uploaded by all the connected users are received, the DBWM scheme immediately launches the weights matching process to find a partner for each connected user.

1) Least Square Distance: Let $FLEs$ denote the maximum number of FL epochs. Let $W_i^{S,k}$ and $W_i^{T,k}$ be the weights of the student and teacher models of U_i at the kth FL epoch, $k = 1, 2, \ldots,$ FLEs. Denote the hidden layers' weights of the student and teacher models of U_i by $W_i^{S_{\text{hidden}},k} \subset W_i^{S,k}$ and $W_i^{T_{\text{hidden}},k} \subset W_i^{T,k}$, respectively. To be specific, $W_i^{S_{\text{hidden}},k}$ consists of the weights of ConvBlock 1, ConvBlock 2, ConvBlock 3, and the dense layer, namely, $W_i^{S_1,k}$, $W_i^{S_2,k}$, $W_i^{S_3,k}$, and $W_i^{S_4,k}$. So, we have $W_i^{S_{\text{hidden}},k} = \{W_i^{S_1,k}, W_i^{S_2,k}, W_i^{S_3,k}, W_i^{S_4,k}\}$.

At the kth FL epoch, user $U_i, i = 1, 2, \ldots, N_{\text{conn}}$, uploads its student model's hidden layers' weights, $W_i^{S_{\text{hidden}},k}$, to the server. The server stores the uploaded weights in the weight set $\mathbf{W}$ defined in the following:

$$\mathbf{W} = \left[W_1^{S_{\text{hidden}},k}, W_2^{S_{\text{hidden}},k}, \ldots, W_{N_{\text{conn}}}^{S_{\text{hidden}},k}\right]. \tag{6}$$

The server then calculates the weights' square distance set, d, based on $\mathbf{W}$. d is defined as

$$d = \begin{bmatrix} d_1 \\ d_2 \\ \cdots \\ d_{N_{\text{conn}}} \end{bmatrix} = \begin{bmatrix} d_{1,2} & \cdots & d_{1,N_{\text{conn}}} \\ d_{2,1} & \cdots & d_{2,N_{\text{conn}}} \\ \cdots & \cdots & \cdots \\ d_{N_{\text{conn}},1} & \cdots & d_{N_{\text{conn}},N_{\text{conn}}-1} \end{bmatrix} \tag{7}$$

where, $d_{i,j}$ $(i, j \in 1, \ldots, N_{\text{conn}}, i \neq j)$ is the square distance between $W_i^{S_{\text{hidden}},k}$ and $W_j^{S_{\text{hidden}},k}$, as defined in the following:

$$\begin{aligned} d_{i,j} &= \left\| W_i^{S_{\text{hidden}},k} - W_j^{S_{\text{hidden}},k} \right\|^2 \\ &= \sum_{m=1}^{4} \left\| W_i^{S_m,k} - W_j^{S_m,k} \right\|^2. \end{aligned} \tag{8}$$

We adopt the argmin function to return the index of the smallest distance for each row in d and obtain the index set, **ID**. **ID** is defined in the following:

$$\mathbf{ID} = \text{argmin}(d) = \left[ID_1, ID_2, \ldots, ID_{N_{\text{conn}}}\right] \tag{9}$$

where, ID_i is the index of the smallest distance for U_i.

Algorithm 1 EFDLS User Implementation Procedure

```
1: procedure USERPROCEDURE($U_i$, $FLEs$)
2:     Initialize all global variables;
3:     for $k = 1$ to $FLEs$ do
4:         if $k == 1$ then
5:             // The student model is trained alone
6:             Obtain $W_i^{S,k}$ after the initial local training;
7:             // Upload its hidden layers' weights to server
8:             Upload $W_i^{S_{hidden},k} \subset W_i^{S,k}$;
9:         else
10:            if receiveServer(Active)==1 then
11:                // Connect to the EFDLS server
12:                Receive $W_i^{LSD,k}$;
13:                Load $W_i^{LSD,k}$ to the teacher model;
14:                Compute $\mathcal{L}_i$ by Eq. (5);
15:                Update $W_i^{S,k+1}$ using the gradient decent;
16:                Upload $W_i^{S_{hidden},k+1} \subset W_i^{S,k+1}$;
17:            else
18:                Disconnect from the EFDLS server.
19:            end if
20:        end if
21:    end for
22: end procedure
```

Based on **ID**, we easily obtain the LSD weight set, $\mathbf{W}^{LSD}$, from $\mathbf{W}$. $\mathbf{W}^{LSD}$ is defined in the following:

$$\begin{aligned} \mathbf{W}^{\text{LSD}} &= \left[W_1^{\text{LSD},k}, W_2^{\text{LSD},k}, \ldots, W_{N_{\text{conn}}}^{\text{LSD},k}\right] \\ &= \left[\mathbf{W}(ID_1), \mathbf{W}(ID_2), \ldots, \mathbf{W}\left(ID_{N_{\text{conn}}}\right)\right] \end{aligned} \tag{10}$$

where, $W_i^{\text{LSD},k}$ are the weights matched with those of U_i at the kth FL epoch.

Once U_i receives $W_i^{\text{LSD},k}$ from the server, U_i loads these weights to its teacher's hidden layers at the beginning of the next FL epoch, as defined in the following:

$$W_i^{T_{\text{hidden}},k+1} \leftarrow W_i^{\text{LSD},k}. \tag{11}$$

Algorithms 1 and 2 show the user and server implementation procedures, respectively.

D. Communication Overhead

EFDLS does not launch the DBWM scheme unless the weights from all the N_{conn} connected users are received. It helps reduce the interaction between the server and users, promoting the system's service efficiency. For user U_i, $i = 1, 2, \ldots, N_{\text{conn}}$, we analyze the communication overhead of uploading and downloading its weights. Denote the bandwidth requirement for uploading the student model's hidden layers' weights of U_i once by BW. Clearly, the bandwidth requirement for downloading the student model's hidden layers' weights from the server once is also BW. That is because, for an arbitrary connected user, the weights uploaded to and those downloaded from the server are of the same size, given that each user has exactly the same model structure. At each FL epoch, the bandwidth requirement for user U_i, $i = 1, 2, \ldots, N_{\text{conn}}$ is estimated as $BW + BW = 2BW$.

TABLE I

DETAILS OF 44 SELECTED DATASETS FROM THE UCR 2018. ABBREVIATION: FLOPs–FLOATING POINT OPERATIONS

Scale	Dataset	Train	Test	Class	SeriesLength	Type	Feature Extractor's Parameter	Feature Extractor's FLOPs
Short	Chinatown	20	345	2	24	Traffic	346,626	696,581
	MelbournePedestrian	1194	2439	10	24	Traffic	347,658	698,629
	SonyAIBORobotSur.2	27	953	2	65	Sensor	346,626	696,581
	SonyAIBORobotSur.1	20	601	2	70	Sensor	346,626	696,581
	DistalPhalanxO.A.G	400	139	3	80	Image	346,755	696,837
	DistalPhalanxO.C.	600	276	2	80	Image	346,626	696,581
	DistalPhalanxTW	400	139	6	80	Image	347,142	697,605
	TwoLeadECG	23	1139	2	82	ECG	346,626	696,581
	MoteStrain	20	1252	2	84	Sensor	346,626	696,581
	ECG200	100	100	2	96	ECG	346,626	696,581
	CBF	30	900	3	128	Simulated	346,755	696,837
Medium	DodgerLoopDay	78	80	7	288	Sensor	347,271	697,861
	DodgerLoopGame	20	138	2	288	Sensor	346,626	696,581
	DodgerLoopWeekend	20	138	2	288	Sensor	346,626	696,581
	CricketX	390	390	12	300	Motion	347,916	699,141
	CricketY	390	390	12	300	Motion	347,916	699,141
	CricketZ	390	390	12	300	Motion	347,916	699,141
	FaceFour	24	88	4	350	Image	346,884	697,093
	Ham	109	105	2	431	Spectro	346,626	696,581
	Meat	60	60	3	448	Spectro	346,755	696,837
	Fish	175	175	7	463	Image	347,271	697,861
	Beef	30	30	5	470	Spectro	347,013	697,349
Long	OliveOil	30	30	4	570	Spectro	346,884	697,093
	Car	60	60	4	577	Sensor	346,884	697,093
	Lightning2	60	61	2	637	Sensor	346,626	696,581
	Computers	250	250	2	720	Device	346,626	696,581
	Mallat	55	2345	8	1024	Simulated	347,400	698,117
	Phoneme	214	1896	39	1024	Sensor	351,399	706,053
	StarLightCurves	1000	8236	3	1024	Sensor	346,755	696,837
	MixedShapesRegularT.	500	2425	5	1024	Image	347,013	697,349
	MixedShapesSmallT.	100	2425	5	1024	Image	347,013	697,349
	ACSF1	100	100	10	1460	Device	347,658	698,629
	SemgHandG.Ch2	300	600	2	1500	Spectrum	346,626	696,581
Vary	AllGestureWiimoteX	300	700	10	Vary	Sensor	347,658	698,629
	AllGestureWiimoteY	300	700	10	Vary	Sensor	347,658	698,629
	AllGestureWiimoteZ	300	700	10	Vary	Sensor	347,658	698,629
	GestureMidAirD1	208	130	26	Vary	Trajectory	349,722	702,725
	GestureMidAirD2	208	130	26	Vary	Trajectory	349,722	702,725
	GestureMidAirD3	208	130	26	Vary	Trajectory	349,722	702,725
	GesturePebbleZ1	132	172	6	Vary	Sensor	347,142	697,605
	GesturePebbleZ2	146	158	6	Vary	Sensor	347,142	697,605
	PickupGestureW.Z	50	50	10	Vary	Sensor	347,658	698,629
	PLAID	537	537	11	Vary	Device	347,787	698,885
	ShakeGestureW.Z	50	50	10	Vary	Sensor	347,658	698,629

For U_i, its total communication overhead is in proportion to $2BW \cdot \text{FLEs}$. Hence, the total communication overhead is proportional to $2BW \cdot \text{FLEs} \cdot N_{\text{conn}}$.

IV. PERFORMANCE EVALUATION

This section first introduces the experimental setup and performance metrics and then focuses on the ablation study. Finally, the performance of EFDLS and communication efficiency is evaluated.

A. Experimental Setup

1) Data Description: The UCR 2018 archive is one of the most popular time series repositories with 128 datasets in various application domains [60]. Following the previous work [53], we divide the UCR 2018 archive into four categories with respect to dataset length, namely, "short," "medium," "long," and "vary." The length of a "short" dataset is no more than 200. That of a "medium" one varies from 200 to 500. A "long" one has a length of over 500 while a "vary" one has an indefinite length. Each "vary" dataset has some NaN data, where NaN stands for Not A Number and is one of the common ways to represent the missing value in the data. It is a unique floating-point value and cannot be converted to any other type than float. NaN value is one of the significant challenges in data analysis. The 128 datasets are composed of 41 "short," 32 "medium," 44 "long," and 11 "vary" datasets. Unfortunately, our limited computing resources do not allow us to consider the whole 128 datasets (detailed hardware specifications can be found in Section IV-A2). There were seven algorithms for performance comparison and the average training time on the 128 datasets costs more than 32 h for a single FL epoch. So, we select 11 datasets from each category, resulting in 44 datasets. More details are found in Table I.

2) Implementation Details: Following previous studies [8], [9], [10], [11], [53], we set the decay value of BN to 0.9.

TABLE II
EXPERIMENTAL RESULTS OF DIFFERENT ALGORITHMS ON 44 DATASETS WHEN $N_{\mathrm{CONN}} = 44$ AND $N_{\mathrm{TOT}} = 44$

Dataset	Baseline	FedAvg	FedAvgM	FedGrad	FTL	FTLS	FKD	EFDLS
Chinatown	0.9623	0.2754	0.2754	0.9623	**0.9665**	0.9537	0.9275	0.9478
MelbournePedestrian	0.9139	0.1	0.1	0.7784	0.8486	0.8922	0.9379	**0.9453**
SonyAIBORobotSur.2	0.8961	0.383	0.383	0.8363	0.8688	0.9035	**0.915**	0.8961
SonyAIBORobotSur.1	0.8652	0.5707	0.6619	0.7887	0.8236	0.8702	0.8369	**0.8819**
DistalPhalanxO.A.G	**0.6763**	0.1079	0.1079	0.6187	0.6259	0.6475	0.6691	0.6475
DistalPhalanxO.C.	0.75	0.417	0.6619	0.6776	0.7464	0.7465	**0.7536**	0.7428
DistalPhalanxTW	0.6547	0.1295	0.1295	0.554	0.6259	0.6547	**0.6835**	0.6403
TwoLeadECG	0.7463	0.4996	0.4996	0.7305	0.7287	0.7278	**0.8112**	0.7665
MoteStrain	0.7788	0.5391	0.5391	0.6933	0.7923	**0.8283**	0.8163	0.8203
ECG200	0.86	0.36	0.36	0.8	0.84	0.85	**0.87**	0.85
CBF	0.987	0.3333	0.5911	0.5911	0.973	0.9922	0.9922	**0.9956**
DodgerLoopDay	**0.575**	0.15	0.15	0.3875	0.55	0.525	0.5125	0.5375
DodgerLoopGame	0.6884	0.5217	0.5217	0.6232	**0.7826**	0.7609	0.7609	0.7464
DodgerLoopWeekend	0.8261	0.7391	0.7391	0.7319	0.8841	0.8913	0.913	**0.9203**
CricketX	0.5897	0.0692	0.1371	0.2256	0.5667	0.6128	0.659	**0.6718**
CricketY	0.5051	0.0949	0.1357	0.1949	0.5	0.4949	0.5538	**0.5974**
CricketZ	0.6205	0.0846	0.0846	0.2256	0.5692	0.6	0.6692	**0.7256**
FaceFour	0.6477	0.1591	0.1591	0.4659	0.6591	**0.6932**	**0.6932**	0.6818
Ham	**0.7143**	0.4857	0.4857	0.6762	0.7048	**0.7143**	0.7048	0.6952
Meat	0.8667	0.3333	0.3333	0.7333	0.8333	0.8333	0.9	**0.917**
Fish	0.5657	0.1371	0.1371	0.2857	0.5771	0.6	0.6	**0.6229**
Beef	**0.7667**	0.2	0.2	0.5667	0.7	0.7	0.7	**0.7667**
OliveOil	0.8333	0.167	0.167	0.7	**0.8667**	**0.8667**	0.8333	0.8333
Car	0.5833	0.233	0.233	0.5	0.5667	0.5833	0.5667	**0.6333**
Lightning2	0.7869	0.459	0.459	0.7705	0.7869	**0.8033**	0.7541	0.7869
Computers	0.78	0.5	0.5	0.584	0.688	0.748	0.788	**0.804**
Mallat	0.7446	0.1254	0.1254	0.4141	0.7638	0.7539	0.7906	**0.8299**
Phoneme	0.2231	0.02	0.02	0.1108	0.2147	0.2247	0.2859	**0.2954**
StarLightCurves	0.9534	0.1429	0.1429	0.5062	0.9519	**0.9584**	0.9571	0.9582
MixedShapesRegularT.	0.8586	0.1889	0.1889	0.2223	0.8384	0.8598	0.8643	**0.8907**
MixedShapesSmallT.	0.8029	0.1889	0.1889	0.2421	0.7942	0.8062	0.8318	**0.8388**
ACSF1	0.77	0.1	0.19	0.19	0.82	**0.89**	0.87	0.88
SemgHandG.Ch2	0.7067	0.65	0.65	0.555	0.72	**0.7383**	0.6867	0.72
AllGestureWiimoteX	0.2643	0.1	0.1	0.1371	0.2729	**0.3043**	0.2929	0.2914
AllGestureWiimoteY	0.2585	0.1	0.1	0.1357	**0.3186**	0.3029	0.2529	0.2829
AllGestureWiimoteZ	0.2886	0.1	0.1	0.1343	0.2671	0.29	**0.4014**	0.3786
GestureMidAirD1	0.5538	0.0384	0.0384	0.0923	0.5462	0.5538	0.4615	**0.5769**
GestureMidAirD2	0.4231	0.0384	0.0384	0.0923	0.4154	0.4462	0.4692	**0.5308**
GestureMidAirD3	**0.3**	0.0384	0.0384	0.0923	0.2693	0.2615	0.2231	0.2769
GesturePebbleZ1	0.4419	0.1628	0.1628	0.2558	0.4767	0.4826	**0.5**	0.4883
GesturePebbleZ2	0.4241	0.1519	0.1519	0.2722	0.5126	0.557	**0.6013**	0.5886
PickupGestureW.Z	0.56	0.1	0.1	0.24	0.62	0.6	0.7	**0.74**
PLAID	0.203	0.0615	0.0615	0.0615	0.2198	0.2253	**0.2924**	0.2589
ShakeGestureW.Z	0.92	0.1	0.1	0.1	**0.96**	0.92	**0.96**	**0.96**
Win	4	0	0	0	3	7	10	**18**
Tie	1	0	0	0	**2**	1	1	**2**
Lose	39	44	44	44	39	36	33	**24**
Best	5	0	0	0	5	8	11	**20**
MeanACC	0.6622	0.2377	0.2557	0.4445	0.6604	0.6743	0.6878	**0.7014**
AVG_rank	3.5455	7.5	7.3409	6.0113	3.9204	2.8977	2.6364	**2.1478**

We use the L_2 regularization to avoid overfitting during the training process. Meanwhile, we adopt the AdamOptimizer with Pytorch,[1] where the initial learning rate is set to 0.0001. Our source code is available at `http://github.com/xiaozw1994/EFDLS`.

All experiments were conducted on a desktop with an Nvidia GTX 1080Ti GPU with 11 GB memory and an AMD R5 1400 CPU with 16 G RAM under the Ubuntu 18.04 OS.

[1]`https://pytorch.org/`

B. Performance Metrics

To evaluate FL algorithms' performance, we use three well-known metrics: "win"/"tie"/"lose," mean accuracy (MeanACC), and AVG_rank, all based on the top-1 accuracy. As suggested in [9], [10], [11], [12], [13], [14], [53], and [56], for an arbitrary algorithm, its "win," "tie," and "lose" values indicate on how many datasets it is better than, equal to, and worse than the others, respectively; its "best" value is the summation of the corresponding "win" and "tie" values.

Algorithm 2 EFDLS Server Implementation Procedure

```
1:  procedure SERVERPROCEDURE($N_{tot}$, $N_{conn}$, $FLEs$)
2:      Initialize all global variables;
3:      Set $\mathbf{W} = \varnothing$;
4:      for $k = 1$ to $FLEs$ do
5:          // Run on the server;
6:          Clear and initialize $\mathbf{W}$;
7:          for $i = 1$ to $N_{conn}$ do
8:              // Receive model weights from users;
9:              Receive $W_i^{S_{hidden},k}$;
10:             Include $W_i^{S_{hidden},k}$ in $\mathbf{W}$.
11:         end for
12:         for $i = 1$ to $N_{conn}$ do
13:             Obtain $W_i^{LSD,k}$ based on $\mathbf{W}$ by Eqs. (6)-(10);
14:             Send $W_i^{LSD,k}$ to $U_i$.
15:         end for
16:     end for
17: end procedure
```

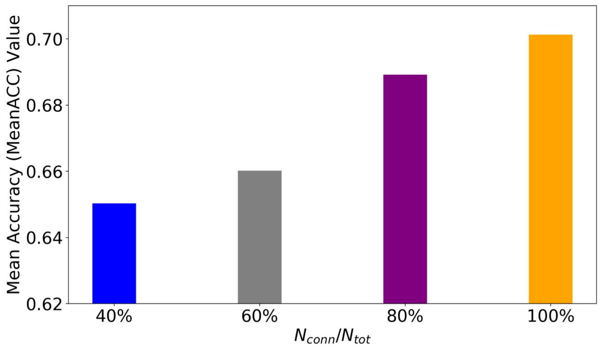


Fig. 2. MeanACC results obtained by EFDLS with different ratios of N_{conn} to N_{tot} on 44 datasets when $N_{\text{tot}} = 44$.

Following [9], [11], [12], [56], and [53], we adopt the AVG_rank score, one of the most widely used robustness tests for ranking various algorithms, where the corresponding results are obtained by the Wilcoxon signed-rank test with Holm's alpha (5%) correction.

C. Ablation Study

We use the 44 UCR2018 datasets above to study the impact of parameter settings on the performance of EFDLS. Assume there are 44 users in the system, i.e., $N_{\text{tot}} = 44$. Each user runs a TSC task with data coming from a specific dataset. For any two users, if they run identical tasks, e.g., motion recognition, their data sources come from different datasets, e.g., CricketX and CricketY. In the experiments, each user's data comes from one of the 44 datasets.

1) Impact of N_{conn}: To investigate the impact of N_{conn} on the EFDLS's performance, we select four ratios of N_{conn} to N_{tot}, namely 40%, 60%, 80%, and 100%. For example, 40% means there are 18 connected users for weights uploading, given $N_{\text{tot}} = 44$. The MeanACC results obtained by EFDLS with different N_{conn} values on 44 datasets are shown in Fig. 2. One can easily observe that a larger N_{conn} tends to result in a higher MeanACC value. That is because as N_{conn} increases, more time series data is made use of by the system, and thus more discriminate representations are captured.

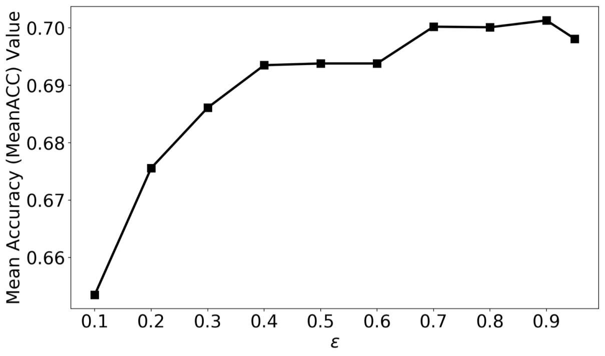


Fig. 3. MeanACC results with different ϵ values on 44 datasets when $N_{\text{conn}} = 44$ and $N_{\text{tot}} = 44$.

2) Impact of ϵ: ϵ is a coefficient to balance each connected user's supervised and KD losses in EFDLS. Fig. 3 shows the MeanACC results with different ϵ values when $N_{\text{conn}} = 44$ and $N_{\text{tot}} = 44$. It is seen that $\epsilon = 0.90$ results in the highest MeanACC score, i.e., 0.7014. That means $\epsilon = 0.90$ is appropriate to reduce each user's entropy on its data during training.

D. Experimental Analysis

To evaluate the overall performance of EFDLS, we compare it with seven benchmark algorithms listed below against "Win"/"Lose"/"Tie," MeanACC, and AVG_rank.

1) *Baseline:* The single-task TSC algorithm with the feature extractor in Fig. 1 deployed on each user. Note that each user has a unique dataset to run and knowledge sharing among the users is disabled.
2) *FedAvg:* The FederatedAveraging method using the feature extractor in Fig. 1 [18].
3) *FedAvgM:* The modified FedAvg using the feature extractor in Fig. 1 [27].
4) *FedGrad:* The federated gradient method using the feature extractor in Fig. 1 [16].
5) *FTL:* The FTL method using the feature extractor in Fig. 1 [23].
6) *FTLS:* FTL [23] based on the DBWM scheme using the feature extractor in Fig. 1.
7) *FKD:* The federated knowledge distillation using the feature extractor in Fig. 1 [27], [28]. For fair comparison, FKD uses the same student–teacher network structure as EFDLS.

Table II shows the top-1 accuracy results with various algorithms on 44 UCR2018 datasets when $N_{\text{conn}} = 44$ and $N_{\text{tot}} = 44$. To visualize the differences between EFDLS and the others, Fig. 4 depicts the accuracy plots of EFDLS against each of the remaining algorithms on 44 datasets. In addition, the AVG_rank results are shown in Fig. 5.

First of all, we study the effectiveness of *knowledge sharing among users* by comparing EFDLS with Baseline. One can observe that EFDLS beats Baseline in every aspect, including

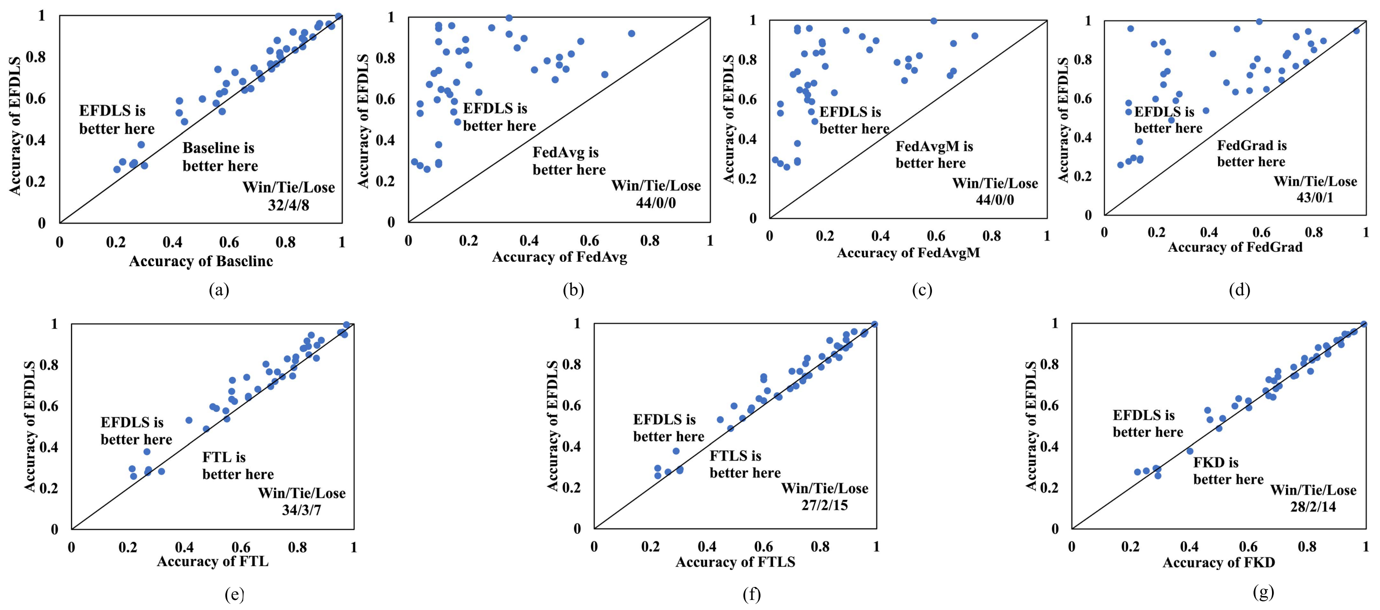


Fig. 4. Accuracy plot results reflecting the performance difference between two given algorithms. (a) EFDLS versus Baseline. (b) EFDLS versus FedAvg. (c) EFDLS versus FedAvgM. (d) EFDLS versus FedGrad. (e) EFDLS versus FTL. (f) EFDLS versus FTLS. (g) EFDLS versus FKD.

"Win"/"Lose"/"Tie," MeanACC, and AVG_rank. For example, the former wins 18 out of 44 datasets while the latter wins only 4. The accuracy plot of EFDLS versus Baseline in Fig. 4(a) also supports the finding above. The main difference between EFDLS and Baseline is that the latter only uses standalone feature extractors which do not share the locally collected knowledge with each other. On the other hand, with sufficient knowledge sharing of similar expertise among the connected users, EFDLS improves the system's generalization ability and thus achieves promising multitask TSC performance.

Second, we study the effectiveness of *the FBST framework* by comparing EFDLS with FTLS. It is easily seen that EFDLS outperforms FTLS regarding the "best," MeanACC, and AVG_rank values. The accuracy plot of EFDLS versus FTLS in Fig. 4(f) also supports this. The FBST framework allows efficient knowledge transfer from teacher to student, helping the student capture sufficient discriminate representations from the input data. On the contrary, the FTLS's learning model lacks self-generalization, leading to deteriorated performance during knowledge sharing.

Third, we study the effectiveness of *the DBWM scheme* by comparing EFDLS with FKD. Apparently, EFDLS overweighs FKD with respect to "best," MeanACC, and AVG_rank. It is backed by the accuracy plot of EFDLS versus FTLS in Fig. 4(g). As mentioned before, at each FL epoch, the DBWM scheme finds a partner for each user and then EFDLS offers weights exchange between each pair of connected users, which realizes knowledge sharing of similar expertise among different users. In contrast, FKD adopts the average weights to supervise the feature extraction process in each user. It is likely to lead to catastrophic forgetting in a user whose weights significantly differ from the average weights.

Last but not least, we compare EFDLS with all the seven algorithms. One can easily observe that our EFDLS is no doubt the best among all algorithms for comparison since ours obtains the highest MeanACC and "best" values, namely 0.7014 and 20, and the smallest AVG_rank value, namely 2.1478. The FKD takes the second position when considering its "best," MeanACC, and AVG_rank values, namely, 11, 0.6878, and 2.6364. On the other hand, FedAvg and its variant, FedAvgM, are the two worst algorithms. The following explains the reasons behind the findings above. When faced with the multitask TSC problem, each user runs one TSC task, and different users may run different TSC tasks. The FBST framework and the DBWM scheme help EFDLS to realize fine-grained knowledge sharing between any pair of users with the most similar expertise. FKD uses the average of all users' weights to guide each user to capture valuable features from the data, promoting coarse-grained knowledge sharing among users. On the other hand, FedAvg and FedAvgM simply take the average weights of all users as each user's weights, which may cause catastrophic forgetting and hence poor performance on multitask TSC.

TABLE III
ACCELERATION PERFORMANCE OF VARIOUS FL ALGORITHMS ON 44 DATASETS ACCORDING TO FEDAVG WHEN $N_{\text{CONN}} = 44$ AND $N_{\text{TOT}} = 44$

Method	FedAvg	FedGrad	FTL	FTLS	FKD	EFDLS
Acceleration	1.0000×	0.9383×	0.8942×	0.8296×	0.7335×	**0.6895**×

E. Communication Efficiency

For each user, the number of hidden layers' parameters of its feature extractor is 346 368. Assume each parameter is a float-type value requiring a space of 4 bytes to store. If we want the parameters to be uploaded (or downloaded) completely within one second, the upload (or download) bandwidth requirement, BW, is calculated as $346\,368 \times 4 \times 8 = 11\,083\,776$ b/s ≈ 11 Mb/s. In this article, we ignore the packet

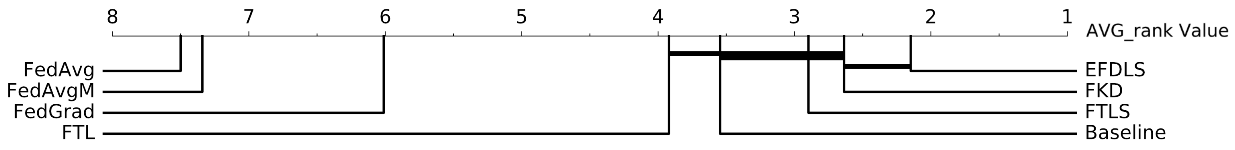


Fig. 5. AVG_rank results of various FL algorithms on 44 datasets.

headers at transport, network and link layers as they are trivial compared with the payload per packet. When $N_{\text{conn}} = 44$, the total upload or download bandwidth requirement of EFDLS is $BW \times N_{\text{conn}}$, namely, $11 \times 44 = 484$ Mb/s, at each FL epoch. Thus, the total bandwidth requirement of EFDLS is $484 \times 2 = 968$ Mb/s after each iteration.

To study the communication efficiency of EFDLS, we compare it with FedAvg, FedGrad, FTL, FTLS, and FKD. Like the previous work [61], we calculate all algorithms' acceleration values based on FedAvg. Table III shows the acceleration results of various FL algorithms on 44 datasets. One can easily observe that the proposed EFDLS has the slowest rate among all algorithms. This is because, unlike the average weights' method, EFDLS uses the DBWM scheme deployed on the server to find the one with the most similar expertise (i.e., a partner) for each user according to LSD. Compared with non-KD methods, FBST deployed on each user consumes additional computational resources to transfer the knowledge from the teacher to its student.

V. Conclusion

The FBST framework promotes knowledge transfer from a teacher's to its student's hidden layers, helping the student capture instance-level representations from the input. The DBWM scheme finds a partner for each user in terms of similarity between their uploaded weights, enabling knowledge sharing of similar expertise among different users. With FBST and DBWM, the proposed EFDLS securely shares knowledge of similar expertise among different tasks for multitask time series classification. Experimental results show that compared with six benchmark FL algorithms, EFDLS is a winner on 44 datasets with respect to the MeanACC and AVG_rank metrics and on 20 datasets in terms of the "best" measure. In particular, compared with the single-task Baseline, EFDLS obtains 32/4/8 regarding the "win"/"tie"/"lose" metric. That reflects the potential of EFDLS to be applied to multitask TSC problems in various real-world domains. We plan to validate EFDLS on more real-world datasets collected from various instruments in the future.

Acknowledgment

The authors sincerely thank the editors and reviewers for their valuable comments and suggestions that greatly improve the quality of the article.

References

[1] N. Yan, L. Zhu, H. Yang, N. Li, and X. Zhang, "Online yarn breakage detection: A reflection-based anomaly detection method," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.

[2] J. Li, H. He, H. He, L. Li, and Y. Xiang, "An end-to-end framework with multisource monitoring data for bridge health anomaly identification," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–9, 2021.

[3] H. Zhu, R. Xiao, J. Zhang, J. Liu, C. Li, and L. Yang, "A driving behavior risk classification framework via the unbalanced time series samples," *IEEE Trans. Instrum. Meas.*, vol. 71, pp. 1–12, 2022.

[4] H. Tong and J. Zhu, "New peer effect-based approach for service matching in cloud manufacturing under uncertain preferences," *Appl. Soft Comput.*, vol. 94, no. 1, pp. 1–17, 2020.

[5] A. Bablani, D. R. Edla, V. Kupilli, and R. Dharavath, "Lie detection using fuzzy ensemble approach with novel defuzzification method for classification of EEG signals," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–13, 2021.

[6] B. M. Maweu, R. Shamsuddin, S. Dakshit, and B. Prabhakaran, "Generating healthcare time series data for improving diagnostic accuracy of deep neural networks," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–15, 2021.

[7] R. M. Mehmood, H.-J. Yang, and S.-H. Kim, "Children emotion regulation: Development of neural marker by investigating human brain signals," *IEEE Trans. Instrum. Meas.*, vol. 70, pp. 1–11, 2021.

[8] H. I. Fawaz, G. Forestier, J. Weber, L. Idoumghar, and P.-A. Müller, "Deep learning for time series classification: A review," *Data Mining Knowl. Discovery*, vol. 33, no. 4, pp. 917–963, Jul. 2019.

[9] H. Fawaz *et al.*, "Time series classification from scratch with deep neural networks: A strong baseline," in *Proc. IEEE IJCNN*, May 2017, pp. 1578–1585.

[10] 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.*, vol. 34, no. 4, Apr. 2020, pp. 6845–6852.

[11] F. Karim, S. Majumdar, H. Darabi, and S. Harford, "Multivariate LSTM-FCNs for time series classification," *Neural Netw.*, vol. 116, pp. 237–245, Aug. 2019.

[12] 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, Sep. 2021.

[13] G. Li, B. Choi, J. Xu, S. Bhowmick, K.-P. Chun, and G. Wong, "ShapeNet: A shapelet-neural network approach for multivariate time series classification," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 9, 2021, pp. 8375–8383.

[14] D. Lee, S. Lee, and H. Yu, "Learnable dynamic temporal pooling for time series classification," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 9, 2021, pp. 8288–8296.

[15] B. Arcas *et al.* (2017). *Federated Learning: Collaborative Machine Learning Without Centralized Training Data*. [Online]. Available: https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[16] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019.

[17] Q. Li *et al.*, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Trans. Knowl. Data Eng.*, early access, Nov. 2, 2021, doi: 10.1109/TKDE.2021.3124599.

[18] M. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, vol. 54, 2017, pp. 1273–1282.

[19] J. Ma, Q. Zhang, J. Lou, L. Xiong, and J. Ho, "Communication efficient federated generalized tensor factorization for collaborative health data analytics," in *Proc. 30th Web Conf.*, 2021, pp. 171–182, 2021.

[20] B. Liu, Y. Guo, and X. Chen, "PFA: Privacy-preserving federated adaptation for effective model personalization," in *Proc. 30th Web Conf.*, Apr. 2021, pp. 923–934.

[21] J. Wu *et al.*, "Hierarchical personalized federated learning for user modeling," in *Proc. 30th Web Conf.*, 2021, pp. 957–968.

[22] Q. Yang, J. Zhang, W. Hao, G. Spell, and L. Carin, "Flop: Federated learning on medical datasets using partial networks," in *Proc. ACM KDD*, 2021, pp. 3845–3853.

[23] Y. Liu, Y. Kang, C. Xing, T. Chen, and Q. Yang, "A secure federated transfer learning framework," *IEEE Intell. Syst.*, vol. 35, no. 4, pp. 70–82, Jul./Aug. 2020.

[24] H. Yang, H. He, W. Zhang, and X. Cao, "FedSteg: A federated transfer learning framework for secure image steganalysis," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1084–1094, Apr. 2021.

[25] D. Dimitriadis, K. Kumatani, R. Gmyr, Y. Gaur, and S. E. Eskimez, "Federated transfer learning with dynamic gradient aggregation," 2020, *arXiv:2008.02452*.

[26] U. Majeed, S. S. Hassan, and C. S. Hong, "Cross-silo model-based secure federated transfer learning for flow-based traffic classification," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2021, pp. 588–593.

[27] H. Seo, J. Park, S. Oh, M. Bennis, and S.-L. Kim, "Federated knowledge distillation," 2020, *arXiv:2011.02367*.

[28] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large CNNs at the edge," in *Proc. NIPS*, 2020, pp. 1–13.

[29] R. Mishra, H. P. Gupta, and T. Dutta, "A network resource aware federated learning approach using knowledge distillation," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, May 2021, pp. 1–2.

[30] S. Itahara, T. Nishio, Y. Koda, M. Morikura, and K. Yamamoto, "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-IID private data," *IEEE Trans. Mobile Comput.*, early access, Mar. 31, 2022, doi: 10.1109/TMC.2021.3070013.

[31] Y. Chen, X. Sun, and Y. Jin, "Communication-efficient federated deep learning with layerwise asynchronous model update and temporally weighted aggregation," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 10, pp. 4229–4238, Oct. 2020.

[32] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Trans. Neural Netw. Learn. Syst.*, vol. 31, no. 9, pp. 3400–3413, Sep. 2020.

[33] L. Nagalapatti and R. Narayanam, "Game of gradients: Mitigating irrelevant clients in federated learning," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 10, 2021, pp. 9046–9054.

[34] X. Cao, J. Jinyuan, and N. Z. Gong, "Provably secure federated learning against malicious clients," in *Proc. AAAI Conf. Artif. Intell.*, vol. 35, no. 8, 2020, pp. 6885–6893.

[35] J. Hong, Z. Zhu, S. Yu, Z. Wang, H. Dodge, and J. Zhou, "Federated adversarial debiasing for fair and transferable representations," in *Proc. ACM KDD*, Aug. 2021, pp. 617–627.

[36] P. Zhou, L. Wang, L. Guo, S. Gong, and B. Zheng, "A privacy-preserving distributed contextual federated online learning framework with big data support in social recommender systems," *IEEE Trans. Knowl. Data Eng.*, vol. 33, no. 3, pp. 824–838, 2021.

[37] Z. Pan, L. Hu, W. Tang, J. Li, Y. He, and Z. Liu, "Privacy-preserving multi-granular federated neural architecture search a general framework," *IEEE Trans. Knowl. Data Eng.*, early access, Sep. 29, 2021, doi: 10.1109/TKDE.2021.3116248.

[38] M. Crawshaw, "Multi-task learning with deep neural networks: A survey," 2020, *arXiv:2009.09796*.

[39] A. P. Ruiz, M. Flynn, and A. Bagnall, "Benchmarking multivariate time series classification algorithms," 2020, *arXiv:2007.13156*.

[40] J. Lines and A. Bagnall, "Time series classification with ensembles of elastic distance measures," *Data Mining Knowl. Discovery*, vol. 29, no. 3, pp. 565–592, Jun. 2015.

[41] A. Bagnall, J. Lines, J. Hills, and A. Bostrom, "Time series classification with cote: The collective of transformation-based ensembles," in *Proc. ICDE*, 2016, pp. 1548–1549.

[42] J. Lines, S. Taylor, and A. Bagnall, "Time series classification with HIVE-COTE: The hierarchical vote collective of transformation-based ensembles," *ACM Trans. Knowl. Discovery Data*, vol. 21, no. 52, pp. 1–35, 2018.

[43] K. Fauvel, É. Fromont, V. Masson, P. Faverdin, and A. Termier, "XEM: An explainable-by-design ensemble method for multivariate time series classification," 2020, *arXiv:2005.03645*.

[44] J. Lines, L. Davis, J. Hills, and A. Bagnall, "A shapelet transform for time series classification," in *Proc. ACM KDD*, 2012, pp. 289–297.

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

[46] A. Dempster, D. F. Schmidt, and G. I. Webb, "MiniRocket: A very fast (almost) deterministic transform for time series classification," in *Proc. 27th ACM SIGKDD Conf. Knowl. Discovery Data Mining*, Aug. 2021, pp. 248–257.

[47] M. G. Baydogan and G. Runger, "Time series representation and similarity based on local autopatterns," *Data Mining Knowl. Discovery*, vol. 30, no. 2, pp. 476–509, Mar. 2016.

[48] J. Large, A. Bagnall, S. Malinowski, and R. Tavenard, "From BOP to BOSS and beyond: Time series classification with dictionary based classifiers," 2018, *arXiv:1809.06751*.

[49] 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. Neural Netw. Learn. Syst.*, vol. 29, no. 4, pp. 920–931, Apr. 2018.

[50] 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, Aug. 2013.

[51] B. Bai, G. Li, S. Wang, Z. Wu, and W. Yan, "Time series classification based on multi-feature dictionary representation and ensemble learning," *Expert Syst. Appl.*, vol. 169, pp. 1–10, May 2021.

[52] H. I. Fawaz *et al.*, "Inceptiontime: Finding AlexNet for time series classification," *Data Mining Knowl. Discovery*, vol. 34, pp. 1936–1962, Sep. 2020.

[53] 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, May 2021.

[54] W. Chen and K. Shi, "Multi-scale attention convolutional neural network for time series classification," *Neural Netw.*, vol. 136, pp. 126–140, Apr. 2021.

[55] S. H. Huang, L. Xu, and C. Jiang, "Artificial intelligence and advanced time series classification: Residual attention net for cross-domain modeling," in *Fintech With Artificial Intelligence, Big Data, and Blockchain* (Blockchain Technologies). Singapore: Springer, 2021, doi: 10.1007/978-981-33-6137-9_5.

[56] Z. Xiao, X. Xu, H. Xing, R. Qu, F. Song, and B. Zhao, "RNTS: Robust neural temporal search for time series classification," in *Proc. IJCNN*, 2021, pp. 1–8.

[57] 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.*, early access, pp. 1–28, 2022, doi: 10.1002/int.22957.

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

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

[60] H. A. Dau *et al.*, "The UCR time series archive," *IEEE/CAA J. Automatica Sinica*, vol. 6, no. 6, pp. 1293–1305, Nov. 2019.

[61] Q. Li, B. He, and D. Song, "Model-contrastive federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Jun. 2021, pp. 10708–10717.

Huanlai Xing (Member, IEEE) received the Ph.D. degree in computer science from the University of Nottingham, Nottingham, U.K., in 2013.

He is an Associate Professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. His research interests include representation learning, data mining, reinforcement learning, machine learning, network function virtualization, and software-defined networking.

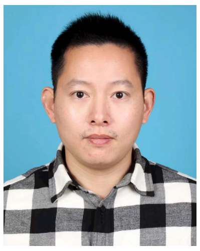
Zhiwen Xiao (Member, IEEE) received the B.Eng. degree in network engineering from the Chengdu University of Information Technology, Chengdu, China, in 2019. He is doing his internship at Southwest Jiaotong University (SWJTU), Chengdu, where he will pursue the Ph.D. degree in computer science.

His research interests include deep learning, federated learning (FL), representation learning, data mining, and computer vision.

Rong Qu (Senior Member, IEEE) received the B.Sc. degree in computer science and its applications from Xidian University, Xi'an, China, in 1996, and the Ph.D. degree in computer science from the University of Nottingham, Nottingham, U.K., in 2003.

She is an Associate Professor at the School of Computer Science, University of Nottingham. Her research interests include the modeling and optimization for logistics transport scheduling, personnel scheduling, network routing, portfolio optimization and timetabling problems 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 has been the Vice-Chair of the Evolutionary Computation Task Committee since 2019 and the Technical Committee on Intelligent Systems Applications from 2015 to 2018 at the IEEE Computational Intelligence Society. She was a Guest Editor for the Special Issues on the Automated Design of Search Algorithms and Machine Learning of the IEEE TRANSACTIONS ON PATTERN ANALYSIS AND MACHINE INTELLIGENCE and *IEEE Computational Intelligence Magazine*. She was an Associate Editor of *IEEE Computational Intelligence Magazine*, IEEE TRANSACTIONS ON EVOLUTIONARY COMPUTATION, *Journal of Operational Research Society*, and *PeerJ Computer Science*.



Zonghai Zhu received the B.Sc. degree from the Department of Information, Mechanical and Electrical Engineering, Shanghai Normal University, Shanghai, China, in 2010, and the Ph.D. degree from the Department of Computer Science and Engineering, East China University of Science and Technology, Shanghai, in 2021.

He is currently an Assistant Professor with the School of Computing and Artificial Intelligence, Southwest Jiaotong University, Chengdu, China. His research interests include imbalanced problems, kernel-based methods, and graph-structured data.

Bowen Zhao received the B.Eng. degree in computer science and technology from Southwest Jiaotong University, Chengdu, Sichuan, China, in 2020, where he is currently pursuing the master's degree with the School of Information Science and Technology.

His research interests include deep reinforcement learning, cloud computing, and deep learning.