Distributed Semi-Supervised Learning With Consensus Consistency on Edge Devices

Hao-Rui Chen, Lei Yang, Xinglin Zhang, *Member, IEEE*, Jiaxing Shen, and Jiannong Cao, *Fellow, IEEE*

***Abstract*—Distributed learning has been increasingly studied in edge computing, enabling edge devices to learn a model collaboratively without exchanging their private data. However, existing approaches assume the private data owned by edge devices are all labeled while the reality is that massive private data are unlabeled and remain to be utilized, which leads to suboptimal performance. To overcome this limitation, we study a new practical problem, Distributed Semi-Supervised Learning (DSSL), to learn models collaboratively with mixed private labeled and unlabeled data on each device. We also propose a novel method *DistMatch* that exploits private unlabeled data by self-training on each device with the help of models from neighboring devices. DistMatch generates pseudo-labels for unlabeled data by properly averaging the predictions of these received models. Furthermore, to avoid self-training with wrong pseudo-labels, DistMatch proposes a *consensus consistency* loss to filter pseudo-labels with high consensus and force the output of the trained model to be consistent with these pseudo-labels. Extensive evaluation results via our self-developed testbed indicate the proposed method outperforms all baselines on commonly used image classification benchmark datasets.**

***Index Terms*—Consistency regularization, distributed machine learning, semi-supervised learning.**

I. Introduction

NOWADAYS, machine learning applications are deployed on many edge devices (such as mobile phones or IoT devices) to provide artificial intelligence services. As a large amount of data are generated at the edge, these applications learn models collaboratively without exchanging private data in a distributed network. Since data labeling could be time-consuming and costly, these generated data on edge devices are largely unlabeled. Therefore, it is important to exploit massive unlabeled data to augment distributed model learning in edge computing.

In a traditional stand-alone computing environment, the model can be trained with Semi-Supervised Learning (SSL) algorithms to improve the model's performance using large-scale unlabeled data. Recently, researchers have proposed consistency regularization [1], [2], [3], [4], [5] to exploit unlabeled data in semi-supervised learning, which optimizes the model by forcing the consistency of the model output between different perturbation versions of the same unlabeled samples. Self-training [6], [7], [8] is another commonly used technique in SSL algorithms, which uses the model's output of unlabeled samples (also known as pseudo-labels) to optimize the model. These methods have achieved great success in the past stand-alone computing environment. Still, in the distributed computing environment, the accuracy of the models trained by these methods has declined sharply due to the small number of samples on each node [9]. Moreover, due to privacy and other issues, these training samples cannot be transmitted to the server for training, so we need to explore how to implement distributed semi-supervised learning without transferring datasets across the network.

To keep privacy and reduce the transmitting overhead, researchers proposed several approaches for distributed machine learning, such as federated learning [10], [11], [12], distributed SGD [13], [14] and gossip learning [15], [16]. These methods train a local model for each device using data on the device, aggregating the models over the network. Private data are kept on each device without transferring across the network and the model is optimized with global data information. However, these methods assume data generated on edge devices are all labeled which is unrealistic in practice. Due to the labor-intensive and time-consuming nature of data labeling in edge computing, only a small portion of data will be labeled. To exploit unlabeled data to augment the models on edge devices, researchers have done several studies and proposed various methods for distributed semi-supervised training. References [17], [18] consider the scenario that there exists a shared unlabeled dataset across the edge devices and use knowledge distillation to train the model in a distributed manner. These methods reduce the communication overhead by only transferring soft labels but can not be applied in the scenario where each edge device has private unlabeled data. To fully use these private data, Federated Semi-Supervised Learning (FSSL) [9], [19], [20], [21],

Manuscript received 3 April 2023; revised 28 October 2023; accepted 4 December 2023. Date of publication 8 December 2023; date of current version 29 December 2023. This work was supported in part by the National Natural Science Foundation of China under Grants 61972161 and 62372185, in part by Guangdong Basic and Applied Basic Research Foundation under Grant 2020A1515011496, in part by the Guangzhou Basic and Applied Basic Research Foundation under Grant 202201010715, in part by Hong Kong RGC General Research Fund under Grants PolyU 15217919 and PolyU 152133/18, in part by Hong Kong RGC Theme-based Research Scheme (TRS) under Grant T43-513/23-N, in part by HK RGC Research Impact Fund under Grant R5060-19, and in part by Lingnan University, Hong Kong Special Administrative Region, China. Recommended for acceptance by S. Wang. *(Corresponding author: Lei Yang.)*

Hao-Rui Chen and Lei Yang are with the School of Software Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: sechenhr@mail.scut.edu.cn; sely@scut.edu.cn).

Xinglin Zhang is with the School of Computer Science and Engineering, South China University of Technology, Guangzhou 510006, China (e-mail: csxlzhang@scut.edu.cn).

Jiaxing Shen is with the Department of Computing and Decision Sciences, Lingnan University, Hong Kong (e-mail: jxshen.polyu@gmail.com).

Jiannong Cao is with the Department of Computing, The Hong Kong Polytechnic University, Hong Kong (e-mail: csjcao@comp.polyu.edu.hk).

Digital Object Identifier 10.1109/TPDS.2023.3340707

1045-9219 © 2023 IEEE. Personal use is permitted, but republication/redistribution requires IEEE permission.
See https://www.ieee.org/publications/rights/index.html for more information.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

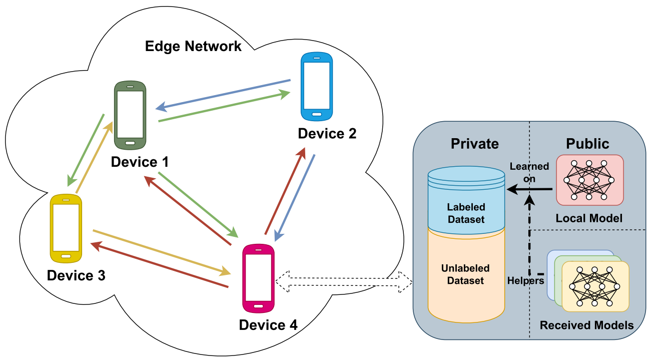


Fig. 1. Illustration of Distributed Semi-Supervised Learning (DSSL). Each device learns the local model on its private dataset consisting of a labeled dataset and an unlabeled dataset, and receives models from its neighboring devices across the network. DSSL aims to learn a model for each device on both the private labeled and unlabeled datasets with the help of the neighbor device's model.

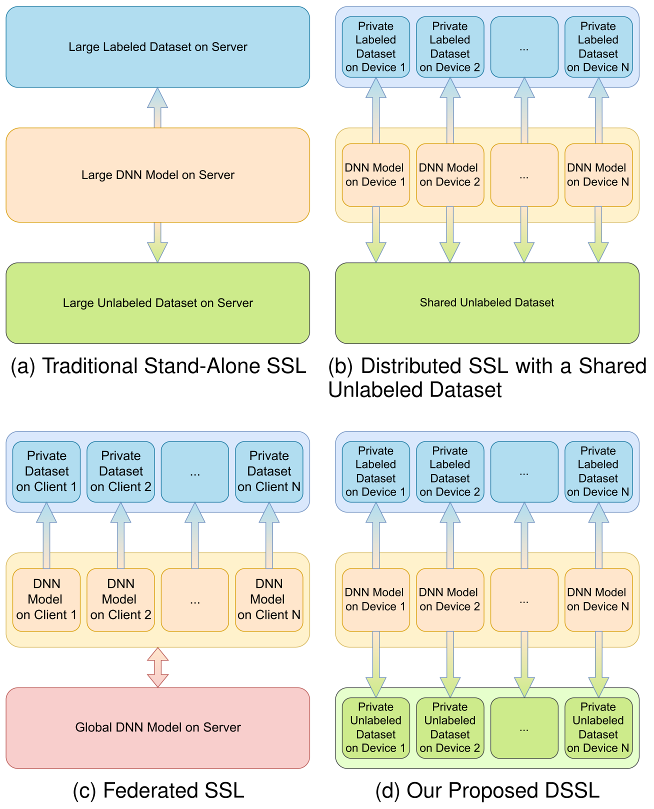

Large Labeled Dataset on Server
Large DNN Model on Server
Large Unlabeled Dataset on Server
(a) Traditional Stand-Alone SSL
Private Labeled Dataset on Device 1
Private Labeled Dataset on Device 2
...
Private Labeled Dataset on Device N
DNN Model on Device 1
DNN Model on Device 2
DNN Model on Device N
Shared Unlabeled Dataset
(b) Distributed SSL with a Shared Unlabeled Dataset
Private Dataset on Client 1
Private Dataset on Client 2
Private Dataset on Client N
DNN Model on Client 1
DNN Model on Client 2
DNN Model on Client N
Global DNN Model on Server
(c) Federated SSL
Private Unlabeled Dataset on Device 1
Private Unlabeled Dataset on Device 2
Private Unlabeled Dataset on Device N
(d) Our Proposed DSSL

Fig. 2. Difference between DSSL and other problem. (a) Traditional SSL trains a model using a large-scale datasets consisting of labeled and unlabeled data. (b) Previous distributed SSL learns models for each device by distilling the logits of the shared unlabeled dataset if exists. (c) Federated SSL learns a global model by local training with client dataset. (d) Our proposed SSL aims at learning a model for each device using the local dataset without shared dataset and centralized server.

[22], [23], [24] is proposed to train a global model with these unlabeled data by self-training or consistency regularization. References [19], [20] combine a distributed learning method and a semi-supervised learning algorithm, where each node trains its model with a semi-supervised learning algorithm, followed by model exchange and aggregation. However, this method only uses an identical global model to predict pseudo-labels for unlabeled samples, which reduces the accuracy of generated pseudo-labels in the data-heterogeneous environment. To tackle this problem, references [9], [21], [22], [23] are proposed to train a global model with pseudo-labels generated by helper models. The aggregation server selects these helper models for each client without the information on the dataset distribution. Therefore, it is difficult for the aggregation server to select proper helper models for each client to generate accurate pseudo-labels. Furthermore, the centralized aggregation server may suffer from a single point of failure, making FSSL unreliable in the edge computing environment.

To address the above issues, we study the practical problem of Distributed Semi-Supervised Learning (DSSL) where each device in the network optimizes its own machine learning model with both labeled data and unlabeled data (See Fig. 1). We show an illustration of the differences between DSSL and other similar problems in Fig. 2. We assume that the owner of each device randomly labels some of the data in the device so that the distribution of the labeled dataset and the unlabeled dataset are similarly identical. At the same time, datasets on different devices are generally not identically and independently distributed (non-IID [12]), which makes it hard for the models trained on different devices to generate high-accuracy pseudo-labels.

To solve the problem of DSSL, we propose a novel framework, **Distributed Matching (DistMatch)**, which leverages private unlabeled data by self-training with the help of the received models from the neighboring devices. Each device validates the received models with its labeled dataset to detect the data distribution similarity between the devices. DistMatch generates pseudo-labels by weighted averaging the predictions of the models from the neighboring devices. The validation result determines the averaging weight of the neighboring model. The model from the neighboring device with a similar data distribution has a higher weight in the averaging. Furthermore, to avoid self-training with wrong pseudo-labels, we extend the consistency regularization in the distributed scenario, and propose a **Consensus Consistency** loss to force models to make identical outputs on the same unlabeled data. We evaluate DistMatch on several benchmark datasets in the IID and the non-IID settings and show that DistMatch outperforms baselines. To sum up, our main contributions are as follows:

- We study the problem of **Distributed Semi-Supervised Learning (DSSL)**, where each device in a network optimizes its machine learning model with the mixed dataset consisting of labeled data and unlabeled data. To the best of our knowedge, we are the first to study the DSSL problem in a fully distributed network without the presence of a centralized server.
- We propose a novel DSSL method, named by **Distributed Matching (DistMatch)**, where each device performs self-training with the consensus consistency loss, and assesses all received models to improve the accuracy of pseudo-labels of unlabeled data.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

- We evaluate our method on several image classification benchmark datasets and show that DistMatch outperforms baselines under both IID and non-IID settings.

The remainder of this article is organized as follows. Section II presents the related work about semi-supervised learning and federated semi-supervised learning. Section III presents the preliminary of distributed semi-supervised learning. Section IV describes DistMatch. Section V presents the experiment we conducted to evaluate DistMatch. Section VI concludes this article.

II. Related Work

A. Semi-Supervised Learning

As numerous approaches are proposed for semi-supervised learning, we only discuss methods closely related to DistMatch. Self-training [6] offers a simple way of SSL by training models with unlabeled data and pseudo-labels generated by the model itself. Obviously, the accuracy of pseudo-labels greatly affects the convergence of the training and the performance of the trained model. Some methods are introduced to reduce the generation of wrong pseudo-labels, such as [3], [25], [26]. Nowadays, self-training has been integrated with other semi-supervised methods, such as consistency regularization [1], [2] and entropy minimization [27]. Consistency regularization was introduced in [1] and assumes that perturbations applied to data would not affect data class semantics [2]. Generally, consistency regularization methods [3], [4], [5], [28], [29], [30], [31], [32] reduce the model output difference between two augmented versions of the same images. References [3], [4], [5], [31], [32] apply weak augmentation on the unlabeled sample to generate the pseudo-labels, and apply strong augmentation to force the consistency with the same unlabeled sample. Recently, FixMatch [3] generates hard pseudo-labels by using a fixed threshold, reaching the state-of-the-art and simplifying semi-supervised methods. Improving upon FixMatch, Dash [4] uses an accending threshold to exploit more unlabeled data. FlexMatch [5] uses a curriculum training method to set dynamic thresholds for each class. Entropy minimization [27] is another popular component for semi-supervised learning, which enforce the model to reduce the entropy of predictions for unlabeled data. Yet, these methods require a considerable amount of unlabeled data to achieve the best accuracy, which is impractical for each edge device. In addition, these methods do not take into account the data heterogeneity on the device, which reduces the performance of these methods.

B. Federated Semi-Supervised Learning

Unlike semi-supervised learning with various methods, Federated Semi-Supervised Learning (FSSL) [33] is a developing field that has attracted the attention of some researchers. From the perspective of dataset partitioning, there may be significant differences in the federated semi-supervised learning problems solved by different FSSL methods. The first dataset partitioning scenario is where each client holds labeled and unlabeled datasets, which was first studied in [19] where each client optimizes the model with the local labeled and unlabeled samples by naive self-training. As knowledge distillation [34] makes transferring knowledge from a big model to a small model possible, some approaches [17], [35] use the server to aggregate the predictions generated by all clients for unlabeled data and deliver aggregated predictions to each client. These methods are usually communication-efficient as the server and clients only transfer the feature and soft labels of the unlabeled data across the network but cannot be applied to the scenario where unlabeled datasets are private. Another way to exploit private unlabeled data on the client is using self-training in local training. There are some works on boosting the accuracy of pseudo-labels for clients, such as multi-view pseudo-labeling [21] for image classification and OmniLabel [36] for federated semi-supervised object detection. Some researchers study FSSL in a different dataset partitioning scenario with more challenges where clients do not have any labeled samples and only the server has a labeled dataset, such as references [9], [20], [22]. Disjoint learning [9] was proposed to learn a model with two parts of parameters on the labeled dataset at the server and the unlabeled datasets at clients, respectively, which mitigates the problem of accuracy reduction in the labels-at-server scenario. SemiFL [20] was proposed with alternate training, where the server fine-tuned the global model with the labeled dataset. Moreover, there is another dataset partitioning scenario where some clients hold labeled datasets and others hold unlabeled datasets [23], [24]. For example, FedSSL [23] proposes a G-Mixup strategy to train a generator that generates artificial data for clients' training without directly using labeled data on other clients, and RSCFed [24] proposed the random sampling consensus FL to average a more robust global model by reweighting each local model trained on different clients. These methods tackled FSSL problems in different scenarios. However, these methods all require a centralized aggregation server to coordinate the training, which presents a single point of failure in FSSL.

Unlike FSSL approaches, our methods are designed as a fully decentralized algorithm for each device, where each device only communicates with its neighboring devices. Each device learns its model collaboratively with the help of received models from neighbors. With the evaluation of the received models, each device detects the data distribution similarity with neighboring devices, mitigating the model accuracy degradation caused by data heterogeneity.

III. Distributed Semi-Supervised Learning

Distributed Semi-Supervised Learning (DSSL) aims to learn a model θ^k collaboratively for each device k connected to a network G. Consider the communication graph $G = (\mathcal{V}, \mathcal{E})$, $\mathcal{V} = \{1, \ldots, N\}$ denotes the devices in the network, and device i can directly communicate with device j if and only if $\{i, j\} \in \mathcal{E}$. Assuming G is connected, device k only communicates its model θ^k with its h-hop neighbors $N(k, h)$.

For the private dataset $\mathcal{D}^k = (\mathcal{D}_s^k, \mathcal{D}_u^k)$ on device k, let $\mathcal{D}_s^k = \{(\mathbf{x}_i^k, y_i^k)\}_{i=1}^{N_s^k}$ denote the labeled dataset where $\mathbf{x}_i^k \in \mathcal{X}$ is a

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

training sample and $y_i^k \in \{1, \ldots, C\}$ is the ground-truth one-hot label corresponding to x_i^k, and $\mathcal{D}_u^k = \{\mathbf{u}_j^k\}_{j=1}^{N_u^k}$ denote the unlabeled dataset, where $|\mathcal{D}_s^k| \ll |\mathcal{D}_u^k|$ generally. We reasonably assume that the two datasets on the same device k follow an identical distribution $\mathcal{P}^k(\mathbf{x})$, while data on different devices are generally not identically and independently distributed (non-IID [12]), i.e. $\mathcal{P}^i \neq \mathcal{P}^j$ for $i \neq j$.

Considering the Deep Neural Network (DNN) model (e.g., ResNet [37]) for C classification problem on device k, we assume that all devices use an identical structure of DNN. Let $p(y|\mathbf{x}, \theta^k) = \sigma(f_{\theta^k}(\mathbf{x}))$ denote the predicted class distribution produced by model θ^k for the input sample $\mathbf{x}$, where

$$\sigma_j(x) = \frac{\exp(x_j)}{\sum_{i=1}^{C} \exp(x_i)} \tag{1}$$

is the softmax function and $f_{\theta^k} : \mathcal{X} \to \mathbb{R}^C$ is the classifier on device k. Each device k randomly initializes its model weights θ^k and learns the model on the dataset $\mathcal{D}^k$ with the objective

$$\min_{\theta^k} \mathcal{L}^k(\theta^k), \tag{2}$$

where the loss function is

$$\mathcal{L}^k(\theta^k) = \mathcal{L}_{sup}^k(\theta^k; \mathcal{D}_s^k) + \lambda_u \mathcal{L}_{uns}^k(\theta^k; \mathcal{D}_u^k). \tag{3}$$

The first term in (3) is the supervised loss term which is the cross-entropy loss generally. The second term is the unsupervised loss term which remains to be determined. To utilize the received models from neighbors $\{\theta^i\}_{i \in N(k,h)}$, each device should validate the received models' knowledge coming from other datasets. As the datasets are private and each device can only learn its model on its dataset, the main challenge of DSSL is to design an algorithm for each device to discover the dataset distribution similarity with its neighbors. With the information on distribution similarity, each device generates pseudo-labels $\hat{y}$ for the unlabeled dataset $\mathcal{D}_u^k$ by properly averaging the predictions generated by the models from neighbors. Finally, each device optimizes its model with the unsupervised loss as the cross-entropy $\mathcal{L}_{uns}^k = \ell_{CE}(\hat{y}, p(y|\mathbf{u}, \theta^k))$. The main notations in this paper are summarized in Table I.

IV. Distributed Matching

In this section, we describe the proposed distributed semi-supervised learning method, namely Distributed Matching (DistMatch).

A. Overall Framework

Overall, all devices in DistMatch follow an identical procedure consisting of two stages: pre-training and self-training. DistMatch requires all devices to synchronize with their neighboring devices after training in each round to ensure that the received models from their neighbors are up-to-date, so that each device will not aggregate the local model with the outdated model coming from a straggler. During the pre-training phase, each device alternately executes two tasks: optimizes the local model using its labeled dataset and aggregates the weights of the local and neighboring models. After pre-training, each device evaluates models received from neighbors and aggregates these received models with the local model. Each device uses received models as helper models to generate pseudo-labels for unlabeled datasets based on the evaluation scores. Then each device updates its model on both labeled and unlabeled datasets. To prevent the use of wrong pseudo-labels, we extend the consistency regularization in the distributed scenario and propose a **Consensus Consistency** loss. Combining with the cross-entropy loss $\mathcal{L}_{CE}$ applied on the labeled dataset, in DistMatch, each device k optimizes its model parameter θ^k using the following loss function:

TABLE I
Summary of Notations

Notation	Explanation
G	The communication graph for all devices
$\mathcal{V}$	The set consists of N devices
$\mathcal{E}$	The set of links connecting the devices
$N(k,h)$	The set of h-hop neighboring devices of device k
θ^k	The model parameters on device k
$\mathcal{D}^k$	The dataset on device k
$\mathcal{D}_s^k$	The labeled dataset on device k
$\mathcal{D}_u^k$	The unlabeled dataset on device k
C	The number of categories
$(\mathbf{x}_i^k, y_i^k)$	The i-th labeled image and its label in the $\mathcal{D}_s^k$
$\mathbf{u}_j^k$	The j-th unlabeled image in the $\mathcal{D}_u^k$
N_s^k	The number of labeled instances in the $\mathcal{D}_s^k$
N_u^k	The number of unlabeled instances in the $\mathcal{D}_u^k$
N_h	The number of models transmitted in the G per round

$$\mathcal{L}^k(\theta^k) = \mathcal{L}_{CE}(\theta^k; \mathcal{D}_s^k) + \lambda_u \mathcal{L}_{CC}(\theta^k; \mathcal{D}_u^k), \tag{4}$$

where the supervised loss $\mathcal{L}_{CE}$ is computed on labeled samples with the weak augmentation function $\alpha(\cdot)$:

$$\mathcal{L}_{CE}(\theta^k; \mathcal{D}_s^k) = \frac{1}{N_s^k} \sum_{(\mathbf{x}^k, y^k) \in \mathcal{D}_s^k} \ell_{CE}(y^k, p(y|\alpha(\mathbf{x}^k), \theta^k)). \tag{5}$$

We present the general process of DistMatch in Algorithm 1 and Fig. 3.

B. Helper Model Evaluation

In DSSL, each device only receives models from its neighboring devices. Thus, the dataset distributions on neighboring devices are unknown to each device. To utilize the models from its neighbors, each device must first evaluate these models to confirm that the knowledge held by these devices can be used to assist the training of the local model. It is clear that models trained on datasets with more similar data distributions will produce pseudo-labels with higher accuracy on the identically and independently distributed unlabeled dataset. However, without transferring the dataset across the network, it is hard to detect the datasets' distribution on neighboring devices. Therefore, we need to find a way to approximately compare the similarity of the datasets distributions on two different devices.

As each device receives models that are trained to fit the datasets on the neighboring devices, if the data distribution of

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

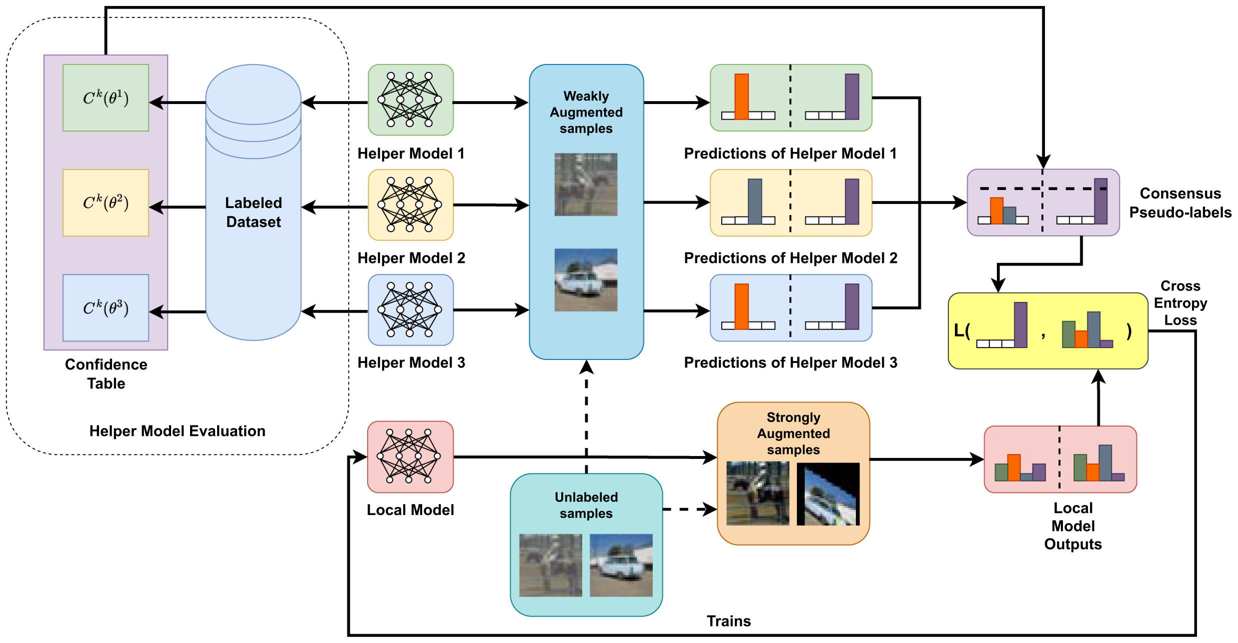


Fig. 3. Overall framework of DistMatch. Each device evaluates its helper models on the local labeled dataset and generates a confidence table for helper models. Then helper models predict the class for the unlabeled sample. These predictions are combined into a consensus pseudo-label according to the confidence table. The pseudo label with a low consensus is eliminated, while the pseudo label with a high consensus is used to compute a cross-entropy loss to train the local model.

the neighboring dataset is quite different from the local dataset, the supervised loss of the model on the local dataset will also be relatively large. So each device can use the supervised loss to distinguish the similarity of the dataset distribution. Since all devices optimize their models with the cross-entropy loss on the labeled datasets, the exponent of the negative cross-entropy loss indicates how confident the model is about the correct class on the labeled data. Thus, each device k computes the supervised loss of the received model coming from up to h-hop neighboring devices on the local labeled dataset and uses the exponent of negative supervised loss as the confidence of the received model output:

$$\mathcal{C}^k(\theta^j) = \exp(-\mathcal{L}^k_{sup}(\theta^j; \mathcal{D}^k_s)), \tag{6}$$

where $j \in N(k,h)$. We name $\mathcal{C}^k$ as the confidence table because it contains the confidence of the correct output for each helper model.

Finally, each device k also exploits its local model to generate pseudo-labels for the unlabeled dataset and compute the confidence of the output $\mathcal{C}^k(\theta^k)$ for its local model. Thus, we denote the set of all helper models for each device k by $\mathcal{H}^k$, which contains the local model θ^k and all received models from up to h-hop neighboring devices. The value of $\mathcal{C}^k(\theta^i)/\mathcal{C}^k(\theta^k)$ can approximately represent the data distribution similarity between $\mathcal{D}^i$ and $\mathcal{D}^k$.

C. Consensus Consistency Loss

Consistency regularization [2] is a component that is currently widely used in the field of semi-supervised learning. Consistency regularization assumes that the class information of the sample will not change after being perturbed. Thus the output of the model to the perturbed sample should be consistent with the original sample. For example, Π-Model [28] optimizes the model with unlabeled data using the following loss:

$$\sum_{i=1}^{\mu B} \| p(y|\pi(u_i), \theta) - p(y|\pi(u_i), \theta) \|_2^2, \tag{7}$$

where $\pi(\cdot)$ is a stochastic augmentation function. After introducing pseudo-labels [6] and strong data augmentation [31], [38], FixMatch [3] uses cross-entropy to optimize the model such as

$$\frac{1}{\mu B} \sum_{i=1}^{\mu B} \mathbb{I}(\max(q) \geq \tau) \ell_{CE}(\hat{q}, p(y|\mathcal{A}(u_i), \theta)), \tag{8}$$

where $\alpha(\cdot)$ and $\mathcal{A}(\cdot)$ denote the weak and strong augmentation functions respectively, $q = p(y|\alpha(u_i), \theta)$ is the output of model for weakly augmented image u_i, $\hat{q} = \arg\max(q)$ is the generated pseudo-label for u_i, and τ is a pre-defined threshold to retain a pseudo-label.

Nevertheless, in DSSL, each device possesses too few data in its local dataset to identify the classes of unlabeled samples by itself. Therefore, in DistMatch, each device also learns pseudo-labels for unlabeled data using the received models from neighboring devices. Instead of directly voting to generate pseudo-labels for the unlabeled dataset, DistMatch believes that models trained on datasets with similar data distributions will produce more accurate pseudo-labels. Since each device k has evaluated all helper models and computed $\mathcal{C}^k(\theta)$ for each helper model $\theta \in \mathcal{H}^k$, it can generate a pseudo-label for each unlabeled sample $\mathbf{u}^k \in \mathcal{D}^k_u$ by properly averaging the predictions of each

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

Algorithm 1: DistMatch: Distributed Matching.

```
Input: the initialized model parameter $\{\theta^i_{(1)}\}_{i=1}^N$ for each device, the number
    of rounds for pre-training $R_p$, the total number of rounds for training $R$, the maximum number of hops for each device to
    obtain neighbor model $h$, the learning rates $\eta$, the batch size of the labeled dataset $B_s$, and the batch size of the unlabeled
    dataset $B_u$.
Output: The trained models $\{\theta^i\}_{i=1}^N$
 1: for round $r = 1$ to $R$ each device $k$ parallel do
 2:   if $r \leq R_p$ then                                                    ▷ Pre-Training Stage
 3:     for each local epoch $e = 1$ to $E$ do
 4:       Randomly split the local labeled dataset $\mathcal{D}^k_s$ into batches $\mathcal{B}^k_s$ of size $B_s$
 5:       for each batch $\{(\mathbf{x}^k_{r,i}, y^k_{r,i})\}_{i=1}^{B_s}$ in $\mathcal{B}^k_s$ do
 6:         Optimize model using ${\theta'}^k_{(r)} = \theta^k_{(r)} - \eta\nabla\mathcal{L}_{CE}(\theta^k_{(r)}; \{(\mathbf{x}^k_{r,i}, y^k_{r,i})\}_{i=1}^{B_s})$
 7:       end for
 8:     end for
 9:   else                                                                   ▷ Self-Training Stage
10:       Generate helper model list $\mathcal{H}^k_{(r)}$ with its confidence table $\mathcal{C}^k_{(r+1)}$
11:       for each local epoch $e = 1$ to $E$ do
12:         Randomly split the local labeled dataset $\mathcal{D}^k_s$ into batches $\mathcal{B}^k_s$ of size $B_s$
13:         Randomly split the local unlabeled dataset $\mathcal{D}^k_u$ into batches $\mathcal{B}^k_u$ of size $B_u$
14:         for each batch $\{(\mathbf{x}^k_{r,i}, y^k_{r,i})\}_{i=1}^{B_s}$ in $\mathcal{B}^k_s$, $\{\mathbf{u}^k_{r,j}\}_{j=1}^{B_u}$ in $\mathcal{B}^k_u$ do
15:           Generate pseudo-labels $\{\hat{y}^k_{r,j}\}_{j=1}^{B_u}$ for $\{\mathbf{u}^k_{r,j}\}_{j=1}^{B_u}$
16:           Optimize model using ${\theta'}^k_{(r)} = \theta^k_{(r)} - \eta\nabla(\mathcal{L}_{CE}(\theta^k_{(r)}; \{(\mathbf{x}^k_{r,i}, y^k_{r,i})\}_{i=1}^{B_s}) + \mathcal{L}_{CC}(\theta^k_{(r)}; \{\mathbf{u}^k_{r,j}\}_{j=1}^{B_u}))$
17:       end for
18:     end for
19:   end if
20:   Synchronize with its $h$-hop neighbors
21:   Exchange models $\{{\theta'}^i_{(r)}\}_{i\in N(k,h)}$ with its $h$-hop neighbors
22:   Aggregate the local model with received models using $\theta^k_{(r+1)} = \frac{\sum_{j\in N'(k,h)}|\mathcal{D}^j|{\theta'}^j_{(r)}}{\sum_{i\in N'(k,h)}|\mathcal{D}^i|}$
23: end for
24: return all trained model parameters $\{\theta^i_{(R+1)}\}_{i=1}^N$
```

helper model as follows:

$$\hat{y}^k = \frac{1}{S^k}\sum_{\theta\in\mathcal{H}^k}\mathcal{C}^k(\theta)\mathbb{I}(\max(q) \geq \tau)\hat{q}, \tag{9}$$

where $q = p(y|\alpha(\mathbf{u}^k), \theta)$ is the output of the helper model θ for the weakly augmented unlabeled instance $\mathbf{u}^k$, $\hat{q}^j = \arg\max(q^j)$ is the predicted one-hot label, τ is a pre-defined threshold to retain the prediction made by the helper model θ, and

$$S^k = \sum_{\theta\in\mathcal{H}^k}\mathcal{C}^k(\theta)\mathbb{I}(\max(q) \geq \tau) \tag{10}$$

is the sum of the helper models' confidence in generating the pseudo-label.

After generating pseudo-labels for unlabeled samples, some of these soft labels may have high entropy. This is because helper models are too divergent when classifying these unlabeled samples. If the device uses these pseudo-labels to train the model, the convergence process of the model training will be greatly disturbed. Therefore, DistMatch uses only those unlabeled samples for which helper models have a high consensus on the classification to promote self-training, resulting in the following **Consensus Consistency** loss:

$$\mathcal{L}_{CC}(\theta^k; \mathcal{D}^k_u) = \frac{1}{N^k_u}\sum_{\mathbf{u}^k\in\mathcal{D}^k_u}\mathbb{I}(\max(\hat{y}^k) \geq \mathcal{T})\ell_{CE}(\hat{y}^k, p^k), \tag{11}$$

where $p^k = p(y|\mathcal{A}(\mathbf{u}^k), \theta^k)$ is the local model output for the strongly augmented unlabeled instance $\mathbf{u}^k$ and $\mathcal{T}$ is a pre-defined consensus threshold to eliminate the pseudo-labels with high entropy. Here we use a strong augmentation function $\mathcal{A}(\cdot)$ to force the consistency between the model outputs and the consensus pseudo-labels.

D. Model Aggregation

After each round of training, each device synchronizes with its h-hop neighboring devices and communicates model parameters across the network. Receiving models from its neighbors, each device aggregates its model parameters with the neighboring models by using the Model Averaging algorithm [39]. We use the size of the dataset on each device as the weights to average received models as below:

$$\theta^k_{(r+1)} = \frac{\sum_{j\in N'(k,h)}|\mathcal{D}^j|\theta^j_{(r)}}{\sum_{i\in N'(k,h)}|\mathcal{D}^i|}. \tag{12}$$

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

Here for convenience, we consider device k to be the 0-hop neighbor of device k, that is $k \in N'(k,h)$. Thus we initialize the local model for the next round of training.

E. Cost Analysis

While DistMatch is proposed to utilize unlabeled data for training models on edge devices, network resources and computing resources are critical in the edge computing environment. Therefore, in this subsection, we briefly analyze the communication and computation costs for DistMatch.

1) Communication Cost: Since DistMatch uses a round-based synchronous communication mechanism, each device only communicates its model weights with their h-hop neighboring devices on the network in each round. Thus, the communication cost of DistMatch is proportional to the number of transmitted models across the network per round. We denote the size of each model by s_t and the number of the transmitted models across the network in each round as N_h. When $h=1$, each device communicates with its directly connected neighboring devices, it is easy to show that the communication cost $C_t(k)$ for each device k is $C_t(k) = 2 * N(k,1) * s_t$. The coefficient 2 in this equation means that each device receives models from its neighboring devices and sends the local model to its neighbors. Noting that $\sum_{i\in\mathcal{V}} N(i,1) = 2*|\mathcal{E}|$, thus in this situation, the total communication cost of the network is $C_t = \frac{1}{2}\sum_{i\in\mathcal{V}} C_t(i) = 2*|\mathcal{E}|*s_t$. When $h \geq 2$, the communication cost of DistMatch is no longer a fixed value about $|\mathcal{E}|$ as the communication cost is affected by the communication topology. As the communication cost $C_t = s_t * N_h$, we approximate the upper bound of N_h as follows:

Theorem 1: In a communication graph $G=(\mathcal{V},\mathcal{E})$, assuming the $\mathcal{M} = \max_{i\in\mathcal{V}} |N(i,1)|$, if each device in $\mathcal{V}$ only communicates its model with its all h-hop neighbors, then the total number of models transferred on the network $N_h \leq 2 * \frac{\mathcal{M}^h - 1}{\mathcal{M}-1}|\mathcal{E}|$.

Proof 1: Note that

$$|N(k,h)| = |N(k,1) \cup \sum_{i\in N(k,1)} N(i,h-1)|. \tag{13}$$

According to the recursion, we have

$$\begin{aligned}
N_h &= 2*\sum_{k\in\mathcal{V}} |N(k,h)| && (14)\\
&= 2*\sum_{k\in\mathcal{V}} |N(k,1) \cup \sum_{i\in N(k,1)} N(i,h-1)| && (15)\\
&\leq 2*\left(|\mathcal{E}| + \sum_{k\in\mathcal{V}}\sum_{k\in N(k,1)} |N(i,h-1)|\right) && (16)\\
&\leq 2*\left[(1+\mathcal{M})|\mathcal{E}| + \sum_{k\in\mathcal{V}}\sum_{i\in N(k,1)}\sum_{j\in N(i,1)} |N(j,h-2)|\right] && (17)\\
&\leq \ldots && (18)\\
&\leq 2*\frac{\mathcal{M}^h - 1}{\mathcal{M}-1}|\mathcal{E}| && (19)
\end{aligned}$$

■

Remark: According to the analysis, we obtain $C_t = O(\mathcal{M}^h)$ when each device communicates with all its h-hop devices. To reduce communication cost, as shown in Section V, DistMatch restricts all devices to only communicating with their one-hop neighbor devices. Note that for some particular network topologies, h has an up-bound less than N. For example, $h \leq 1$ for a fully connected network, $h \leq 2$ for a star topology network. It does not make sense when h exceeds the up-bound.

2) Computation Cost: We discuss the computation time cost for each device. In DistMatch, the computation time cost can be divided into the following three parts: time cost of gradient computation T_g, time cost of model inference T_i, and time cost of model aggregation T_a. Obviously, the gradient computation cost T_g is constant as each device only computes gradients using the loss function 4. For model aggregation time T_a, as each device k aggregates with its all received models the number of which is $N(k,h)$, we obtain that $T_a = O(N(k,h))$. For the term T_i, we only discuss the self-training stage in DistMatch.

To generate the confidence table $\mathcal{C}^k$, each device k use the labeled dataset to validate all its received models. Assuming the computation time for each model to generate the output of each sample is t_{inf} and the time cost to compute the cross-entropy for each output is t_{ce}. Thus the computation time cost for generating $\mathcal{C}^k$ is $N_s^k(t_{inf} + t_{ce})$ for each model. To generate pseudo-labels for $\mathcal{D}_u^k$, the computation time for each model is $N_u^k * t_{inf}$ and the cost of averaging the prediction is $N_u^k * |N(k,h)| * t_{avg}$, where t_{avg} is the time cost for averaging two predicted one-hot labels. To train its local model, the computation cost for forwarding propagation is $(N_s^k + N_u^k) * (t_{inf} + t_{ce})$. As $t_{inf} \gg t_{ce}$ and t_{avg} generally, the total time cost for model inference can be approximated as $(|N(k,h)|+1)(N_s^k + N_u^k)t_{inf}$. According to the analysis of the communication cost, we can show that $|N(k,h)| = O(\mathcal{M}^h)$. Thus limiting the value of h also reduces the computation cost.

V. Experiments

In this section, we first validate DistMatch on several commonly used image classification datasets by comparing its performance with traditional learning approaches and analyzing the impact of the accuracy of pseudo-labels generated for unlabeled data. Then, we conduct ablation experiments to verify the effectiveness of DistMatch components, followed by the impact of other hyper-parameters on DistMatch.

A. Setup

1) Datasets: We validate DistMatch on three image classification datasets that are commonly used in semi-supervised learning evaluations, including Fashion-MNIST [40], CIFAR-10 [41], SVHN [42], and STL-10 [43]. Fashion-MNIST [40] is a balanced image dataset obtained from fashion products on Zalando's website, which consists of 60,000 training instances and 10,000 testing instances. CIFAR-10 [41] is a widely-used balanced object classification dataset that contains 50,000 training instances and 10,000 testing instances across 10 classes. SVHN [42] is a real-world image dataset obtained from house

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

TABLE II
LeNet4 Architecture

Layer	Filter Shape	Stride
Input	N/A	N/A
Conv 1	$5 \times 5 \times 1 \times 4$	1
Average Pool	2×2	2
Conv 2	$5 \times 5 \times 4 \times 6$	1
Average Pool	2×2	2
FC	96×120	N/A
FC	120×10	N/A

TABLE III
ResNet10 Architecture

Layer	Filter Shape	Stride
Input	N/A	N/A
Conv 1	$3 \times 3 \times 3 \times 64$	1
Max Pool	2×2	2
Conv 2	$3 \times 3 \times 64 \times 64$	1
Conv 3	$3 \times 3 \times 64 \times 64$	1
Conv 4	$3 \times 3 \times 64 \times 128$	2
Conv 5	$3 \times 3 \times 128 \times 128$	1
Downsample.Conv	$1 \times 1 \times 64 \times 128$	1
Conv 6	$3 \times 3 \times 128 \times 256$	2
Conv 7	$3 \times 3 \times 256 \times 256$	1
Downsample.Conv	$1 \times 1 \times 128 \times 256$	1
Conv 8	$3 \times 3 \times 256 \times 512$	2
Conv 9	$3 \times 3 \times 512 \times 512$	1
Downsample.Conv	$1 \times 1 \times 256 \times 512$	1
Average Pool	4×4	4
FC	512×10	N/A

numbers in Google Street View images, consisting of 73,257 training instances, 26,032 testing instances, and 202,353 extra instances. STL-10 [43] is another dataset for unsupervised learning, which consists of 5,000 labeled training images, 100,000 unlabeled training images, and 8,000 testing images. As the unlabeled dataset contains data that does not belong to any of the ten categories of the labeled dataset, training on this dataset is closer to the realistic settings and more challenging.

2) Emulation Settings: We conduct the experiments on our self-developed EdgeTB [44], an open-source emulator designed for distributed machine learning in edge computing. EdgeTB can emulate arbitrary edge networks as demanded on high-performance computers, allowing users to configure the resources on edge devices and the network. For each set of experiments, we randomly construct a connected network graph with 100 emulated device nodes and 1000 links in the EdgeTB, i.e., $|\mathcal{V}| = 100$ and $|\mathcal{E}| = 1000$. Besides, to reduce communication costs, we restrict each device to communicating with its one-hop neighbors only.

3) ML Model: Due to edge devices' poor computing power, we use the model architecture LeNet4 [45] for the Fashion-MNIST dataset and ResNet-10 [37] for other datasets in experiments for all baselines and our method. We show the architecture in Table II for LeNet4 and Table III for ResNet10. For LeNet4, the first convolutional neural layers have 4 filters of the same 5×5 kernel sizes followed by 2×2 average-pooling layer. Next, the second convolutional neural layer has 6 filters followed by 2×2 average-pooling layer. Finally, we have a fully-connected layer with 120 units, followed by adding a softmax fully-connected layer as the classifier of the network. For ResNet10, the first convolutional neural layers have 64 filters and the same 3×3 kernel sizes followed by 2×2 max-pooling layer, which is the stem block of the architecture. Then the second and the third convolutional neural layers have 64 and 128 filters and the same 3×3 kernel sizes followed by 2×2 max-pooling layer. We add a skip connection between the two convolution layers with 128 filters. Then, we add a downsample convolution layer with 128 filters. Next, we repeat the previous step but double the filter size, thus, we have the two convolution layers and a downsample layer with 256 filters, respectively. We repeat one more time and we have the two convolution layers and a downsample layer with 512 filters. Finally, we use an average pooling layer to reduce the kernel size from 4×4 to 1×1, followed by adding a softmax fully-connected layer as the classifier of the network. All parameters are initialized by the variance scaling method.

4) Data Preprocessing and Data Partition: Before dividing data into each device, we preprocess each dataset to provide similar data distribution settings. We do not preprocess Fashion-MNIST and CIFAR-10 as they are balanced datasets. However, as the SVHN is an imbalanced dataset across 10 classes, we would like to construct a balanced dataset on SVHN like CIFAR-10, which contains 50,000 training instances and 10,000 testing instances across 10 classes. Therefore, we combine the training set and testing set of the SVHN, and select 5,000 samples per class for training and 1,000 samples per class for testing from this large dataset. From this, we have a balanced SVHN dataset with 50,000 training instances and 10,000 test instances across 10 categories. STL-10 consists of images of 96×96 resolutions. In order to maintain a similar number of model parameters without increasing the computational cost too much, we use a 3×3 average pooling layer to downsample the image, so that the input shape is reduced to 32×32 which is identical to CIFAR-10 and SVHN dataset.

After the dataset preprocessing, we generate each training set for each device. The first step is splitting the dataset into the labeled dataset and the unlabeled dataset with the size of $|\mathcal{D}_s|$ and $|\mathcal{D}_u|$, respectively. As the training set of Fashion-MNIST, CIFAR-10, and SVHN are all labeled, we randomly select $|\mathcal{D}_s|$ samples as the labeled dataset and $|\mathcal{D}_u|$ as the unlabeled dataset using an identical distribution. We fix $|\mathcal{D}_s| = 6000$ and $|\mathcal{D}_u| = 54000$ for Fashion-MNIST and $|\mathcal{D}_s| = 5000$ and $|\mathcal{D}_u| = 45000$ for CIFAR-10 and SVHN in all experiments. For STL-10, we use the total labeled dataset and the unlabeled dataset in all experiments, i.e., $|\mathcal{D}_s| = 4000$ and $|\mathcal{D}_u| = 100000$. Then, according to the data distribution settings, we divide each dataset into each device using the corresponding method. For the IID setting, to generate a labeled training set for each device, we assign each device $|\mathcal{D}_s|/(C \times |\mathcal{V}|)$ labeled samples per class, so each device owns a labeled dataset of size $|\mathcal{D}_s|/|\mathcal{V}|$. To generate unlabeled training sets, we assign each device $|\mathcal{D}_u|/(C \times |\mathcal{V}|)$ unlabeled samples per class for Fashion-MNIST, CIFAR-10 and SVHN, and $|\mathcal{D}_u|/|\mathcal{V}|$ unlabeled samples for STL-10. Thus, for each device $k \in \{1, \ldots, 100\}$, we have $|\mathcal{D}_s^k| = 60$ and $|\mathcal{D}_s^k| = 540$ for Fashion-MNIST, $|\mathcal{D}_s^k| = 50$ and $|\mathcal{D}_s^k| = 450$ for CIFAR-10 and SVHN, and $|\mathcal{D}_s^k| = 40$ and $|\mathcal{D}_s^k| = 1000$ for STL-10. For the non-IID settings, we only use the CIFAR-10 and SVHN as it is difficult to control the unlabeled dataset distribution of STL-10.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

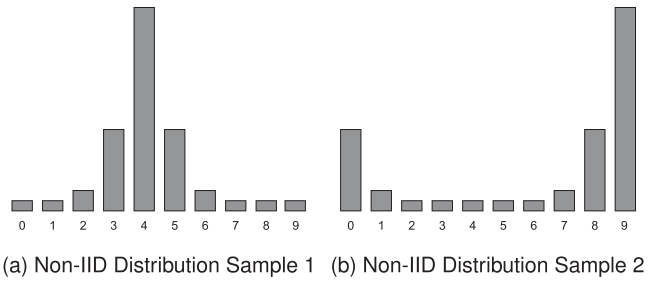


Fig. 4. Samples of non-IID data distribution on a device.

TABLE IV
HYPERPARAMETERS

Hyperparameters	Values
$\mathcal{T}$ (Threshold for filtering pseudo-labels)	0.8
R_p (Rounds for pretraining)	1
h (Communication Hop on Network)	1
λ_u (Coefficient of Unsupervised Loss)	1
B_s (Batch Size)	32
η (Learning Rate)	0.03
β (Momentum)	0.9
Nesterov Momentum	True

We follow the settings in [9] to simulate a class-imbalanced environment but make a few changes. As shown in Fig. 4, datasets on each device are unbalanced, and different datasets on the different devices are non-IID. Unlike the settings in [9], we keep an identical data distribution of the labeled dataset and the unlabeled dataset on the same device. We similarly generate a labeled dataset of size $|\mathcal{D}_s|/|\mathcal{V}|$ and an unlabeled dataset of size $|\mathcal{D}_u|/|\mathcal{V}|$ for each device.

5) Baselines: Our baselines are listed below:

- *Local SGD:* Local supervised learning with only the labeled dataset $\mathcal{D}_s^k$ for each device k.
- *Local FixMatch:* Local semi-supervised learning using the FixMatch algorithm [3] with both the labeled dataset $\mathcal{D}_s^k$ and the unlabeled dataset $\mathcal{D}_u^k$ for each device k, where each device uses a fixed high threshold to generate pseudo-labels.
- *D-SGD:* Distributed SGD [13] with only the labeled dataset $\mathcal{D}_s^k$ for each device k.
- *D-FixMatch:* Naive combination of Distributed SGD and the FixMatch algorithm with both the labeled dataset $\mathcal{D}_s^k$ and the unlabeled dataset $\mathcal{D}_u^k$ for each device k.
- *D-Dash:* Naive combination of Distributed SGD and the Dash algorithm [4] with both the labeled dataset $\mathcal{D}_s^k$ and the unlabeled dataset $\mathcal{D}_u^k$ for each device k, where each device uses an adaptive threshold to generate pseudo-labels.

6) Hyperparameters: We illustrate the parameters used in the experimental section in Table IV. To ensure the fairness of the experiments, we use the same values for all hyperparameters except τ and $\mathcal{T}$ for the same set of experiments. While B_s are shown in Table IV, the unlabeled dataset batch size is computed as $B_u = \frac{N_u^k}{N_s^k} B_s$. We use an identical model architecture, ResNet-10, on each device for all experiments. For weak data augmentation, we use a flip-and-shift strategy, which randomly

TABLE V
TOP-1 ACCURACY COMPARISON IN IID SETTINGS (IN %)

Method	Datasets			
	Fashion-MNIST	CIFAR-10	SVHN	STL-10
Local SGD	52.49	25.17	12.62	23.79
Local FixMatch	51.81	24.78	12.51	23.98
D-SGD	40.70	68.79	60.36	28.02
D-FixMatch	68.92	73.96	88.73	58.87
D-Dash	73.21	68.01	86.29	**63.39**
DistMatch	**75.10**	**76.50**	**89.95**	60.69

TABLE VI
TOP-1 ACCURACY COMPARISON IN NON-IID SETTINGS (IN %)

Method	Datasets		
	Fashion-MNIST	CIFAR-10	SVHN
Local SGD	35.05	17.12	11.78
Local FixMatch	38.06	17.37	11.41
D-SGD	62.97	47.43	70.75
D-FixMatch	66.85	70.69	85.17
D-Dash	73.33	69.86	84.99
DistMatch	**74.50**	**72.34**	**88.14**

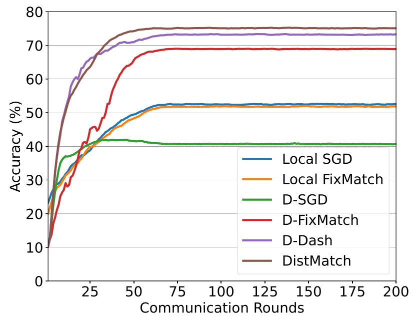
shifts the image horizontally and vertically by 12.5% of its length and width for all datasets, and flip the images left and right with a probability of 50% for all datasets except SVHN. We use RandAugment [38] for strong data augmentation. For FixMatch in baselines, we fix the threshold $\tau = 0.95$ and use the same data augmentation strategies. We use the Stochastic Gradient Descent (SGD) to optimize each model with the learning rate η and the Nesterov momentum β.

B. Performance Results

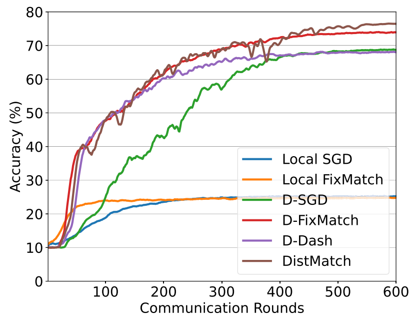
1) Experiment Results on IID Settings: We conduct DistMatch and other baselines in IID settings using four datasets, Fashion-MNIST, CIFAR-10, SVHN, and STL-10. We calculate the mean of the top-1 accuracy of the models on 100 devices after 600 training rounds for datasets CIFAR-10, 200 training rounds for dataset Fashion-MNIST and STL-10, and 150 training rounds for dataset SVHN, and show the results in Table V for IID settings. Fig. 5 shows the change in the mean of validation accuracy over 100 devices during training in IID settings. As these results show, our method achieves the best performance on datasets CIFAR-10 and SVHN. However, as shown in Table V, because of some out-of-distribution data in the unlabeled dataset of STL-10, the performance of models trained by DistMatch on STL-10 is second only to the D-Dash method.

2) Experiment Results on Non-IID Settings: We conduct DistMatch and other baselines in Non-IID settings using three datasets, Fashion-MNIST, CIFAR-10, and SVHN, and show the experimental results in the Table VI and Fig. 6. As these results show, our method achieves the best performance on all datasets in non-IID settings, which means that our method works

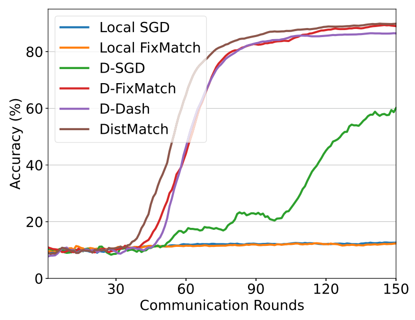
Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.



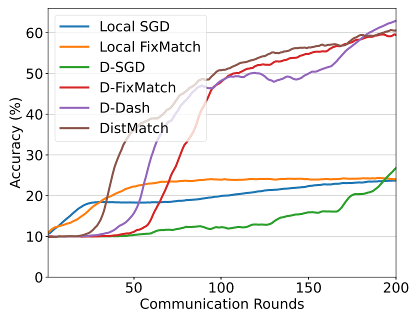
(a) Fashion-MNIST



(b) CIFAR-10

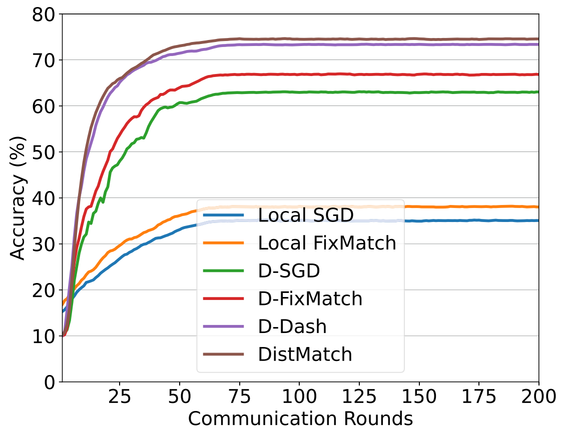


(c) SVHN

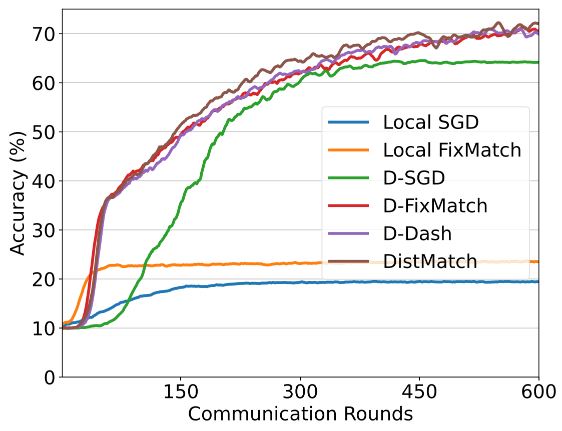


(d) STL-10

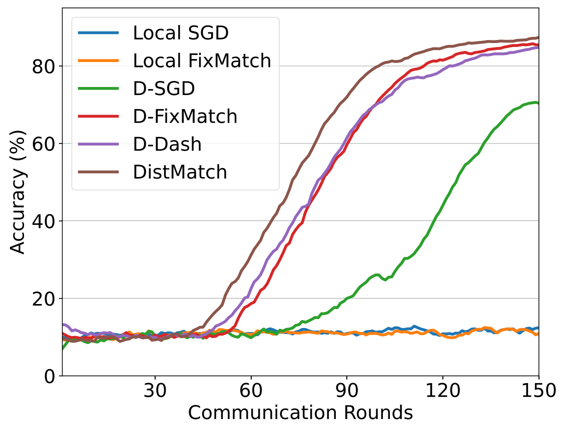
Fig. 5. Average validation accuracy on 100 devices in IID settings.



(a) Fashion-MNIST



(b) CIFAR-10



(c) SVHN

Fig. 6. Average validation accuracy on 100 devices in Non-IID settings.

under all settings. In addition, our approach can achieve higher accuracy using fewer communication rounds. For example, in training the model with the SVHN dataset to achieve 80% accuracy in the non-IID setting, our method uses 98 rounds, while the D-FixMatch method uses 114 rounds.

To explore what makes our method outperform all the baselines in non-IID settings, we conduct experiments to compare the accuracy of pseudo-labels generated by different methods using the dataset CIFAR-10. Table VII shows the results of the number of corrected and wrong-generated pseudo-labels at the end of the training. According to Table VII, DistMatch generates more correct pseudo-labels and fewer wrong pseudo-labels for

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

TABLE VII
ACCURACY OF PSEUDO-LABELS IN NON-IID SETTINGS ON CIFAR-10

Methods	D-FixMatch	DistMatch
Total Correct Pseudo-Labels	33912	32498
Masked Correct Pseudo-Labels	23964	20041
Masked Pseudo-Labels	26364	21538
Masked Pseudo-Labels Accuracy (%)	90.90	93.05
Model Accuracy (%)	70.69	72.34

TABLE VIII
COMPARISON WITH FSSL METHODS ON CIFAR-10 (IN %)

Method	Communication Cost	Distribution	
		IID	Non-IID
FedSem	18.3	49.86	46.22
FedMatch	64.1	55.42	49.36
RSCFed	55.0	46.01	41.87
DistMatch	100.0	**76.50**	**72.34**

unlabeled datasets, which boosts the performance of models on all devices after semi-supervised learning.

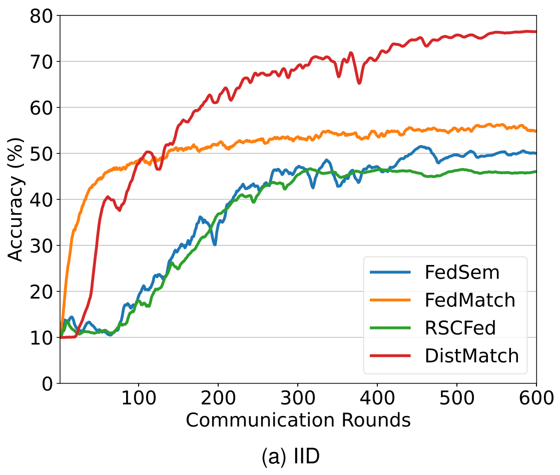
3) Comparasion With FSSL: In FSSL, the dataset settings of the labels-at-server scenario [9] are similar to those in DSSL, where each client trains the local model on both the labeled dataset and unlabeled dataset, and the server averages the received local models and generates the global model for training in next round. Thus, to better illustrate the performance of DistMatch, we conducted experiments to compare DistMatch with FSSL methods. We can arbitrarily specify a device in the network as the server and other devices as the client. Since the server device also holds the dataset, we also consider it as a client. Therefore, we can compare the performance of DistMatch with FSSL methods. We evaluated their performance with 100 edge devices under both IID and Non-IID settings, comparing their model training accuracy and convergence quality at the same number of 600 rounds for CIFAR-10.

As illustrated in Table VIII and Fig. 7, compared to FedSem [19], FedMatch [9] and RSCFed [24], DistMatch achieves a significantly higher accuracy and convergence quality at the same number of 600 rounds, while DistMatch consumes the most communication cost in these methods. This is because FSSL methods only need to transmit the model on a certain spanning tree of the communication graph G, while DistMatch needs to transmit the model on all edges $\mathcal{E}$ of the communication graph G. However, to upload the model from client devices far away from the server device, the model transmission needs to go through a multi-hop network, resulting in higher communication costs for links closer to the server device.

C. Ablation Study

To evaluate the effectiveness of the components in the DistMatch, we conduct ablation studies using CIFAR-10. In this subsection, we evaluate the effectiveness of the helper model evaluation and the consensus consistency loss. We also verified the effect of different values of $\mathcal{T}$ on the accuracy of models. We also discuss the impact on accuracy if the argmax function is applied to the consensus pseudo-labels to generate hard labels. Finally, we use different h in experiments to explore the influence of the number of neighbor nodes on the model accuracy.

1) Effectiveness of the Helper Model Evaluation: To evaluate the effectiveness of the helper model evaluation, we conduct experiments in the Non-IID settings. We eliminate the model evaluation by setting the same value in the confidence table. We show the experimental results in the Table IX. As these data show, the accuracy has slightly dropped from the DistMatch,



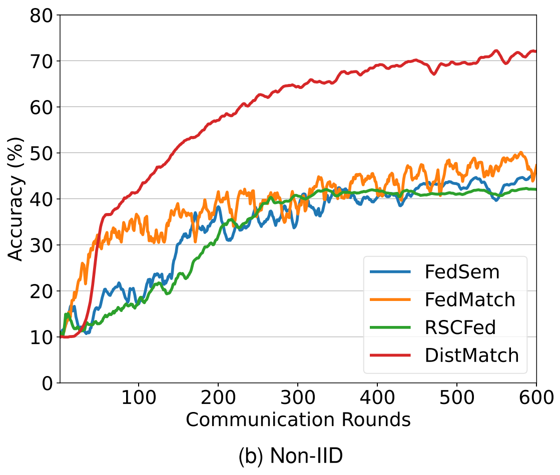


Fig. 7. Average validation accuracy on 100 Devices on CIFAR-10.

TABLE IX
EFFECTIVENESS OF EACH COMPONENTS IN DISTMATCH IN NON-IID SETTINGS ON CIFAR-10

Methods	Average Accuracy (%)
DistMatch	57.78
DistMatch w/o CC Loss	45.52
DistMatch w/o Model Evaluation	56.56
DistMatch with Argmax Function	56.48

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

TABLE X
IMPACT OF $\mathcal{T}$ ON ACURACY IN IID SETTINGS ON CIFAR-10

$\mathcal{T}$	0.4	0.6	**0.8**	0.95
Average Accuracy (%)	61.24	63.51	**64.70**	63.39

TABLE XI
IMPACT OF h ON ACCURACY IN NON-IID SETTINGS ON CIFAR-10

Value of h	Average Accuracy (%)	Time Cost (sec.)
1	57.78	3.07×10^5
2	59.56	3.91×10^6

which means that the helper model evaluation improves the pseudo-labels' accuracy so that DistMatch improves the generalization ability of models learned on these pseudo-labels.

2) Effectiveness of the Consensus Consistency Loss: To evaluate the effectiveness of the consensus consistency loss, we conduct experiments in the Non-IID settings. We eliminate our consensus consistency loss by only training the model with the weakly augmented labeled dataset. In the Table IX, the accuracy has dropped significantly from the DistMatch, which means that the consensus consistency loss helps the model to learn the consistency from the unlabeled dataset with the right knowledge.

3) Effectiveness of Consensus Pseudo-Labels: While our method uses soft pseudo-labels for unlabeled data to optimize models, we conduct experiments in the Non-IID settings to examine whether the soft or hard labels are better. In this experiment, we use an argmax function to generate one-hot pseudo-labels and show the experimental results in the Table IX. As shown in the table, the argmax function reduces the accuracy of DistMatch, implying that the soft labels improve the model's generalization ability as the knowledge distillation.

4) Number of $\mathcal{T}$: As the $\mathcal{T}$ is an important threshold to retain the pseudo-labels, we conduct experiments in Non-IID settings to explore the impact of $\mathcal{T}$ on performance. We try $\mathcal{T} \in \{0.4, 0.6, 0.8, 0.95\}$ and show the experimental results in the Table X. As shown by the results, compared to the lower value and the higher value, the value of 0.8 is a proper parameter for $\mathcal{T}$. The reason for the accuracy reduction is clear: If the $\mathcal{T}$ is too low, models will learn on more unlabeled data with wrong pseudo-labels. Conversely, if the $\mathcal{T}$ is too high, models will learn on less unlabeled data with the right pseudo-labels.

5) Number of h: While all the experiments described previously are performed with each device using only its one-hop neighbors, we also conduct experiments in non-IID settings to examine the influence of the number of neighboring devices on the model accuracy. Let each device in the network fetch models from its h-hop neighbors, where we try $h \in \{1, 2\}$. We show the experimental results in Table XI. We see a 1.78 percentage point improvement in the model's accuracy, which implies that getting helper models from more devices helps improve local model accuracy. However, we also see a 13x increase in time cost as the number of the fetched models increases, which cannot be ignored for each device.

6) Number of Helper Models: While all the experiments described previously are performed with each device using all received models as its helper models, we also conduct experiments in non-IID settings on Fashion-MNIST to examine the influence of the number of helper models on the accuracy of the local model. Let each device randomly select N_r helper models from its received models, where we try $N_r \in \{1, 2, 4, 8, 16, all\}$. We show the experimental results in Table XII and Fig. 8, including the model accuracy and the computation cost for model evaluation T_{me}, pseudo-label generation T_{plg}, model training T_{mt}. As shown by these results, when using fewer helper models, the model accuracy is lower, and $T_{plg} + T_{me}$ is also lower; when the number of helper models used is close to that of all helper models, $T_{plg} + T_{me}$ is longer and the model accuracy is also higher. However, this is also a drawback of DistMatch, which requires a lot of helper models to train the local model to achieve higher accuracy, resulting in relatively high computational overhead.

TABLE XII
IMPACT OF N_r IN NON-IID SETTINGS ON FASHION-MNIST

N_r	1	2	4	8	16	all
T_{me} [sec./dev.]	0.016	0.027	0.046	0.090	0.154	0.220
T_{plg} [sec./(dev.*ep.)]	0.119	0.195	0.371	0.862	1.721	2.036
T_{mt} [sec./(dev.*ep.)]	0.916	1.021	1.226	1.059	1.187	1.103
Accuracy [%]	68.08	71.68	70.31	69.80	74.37	74.50

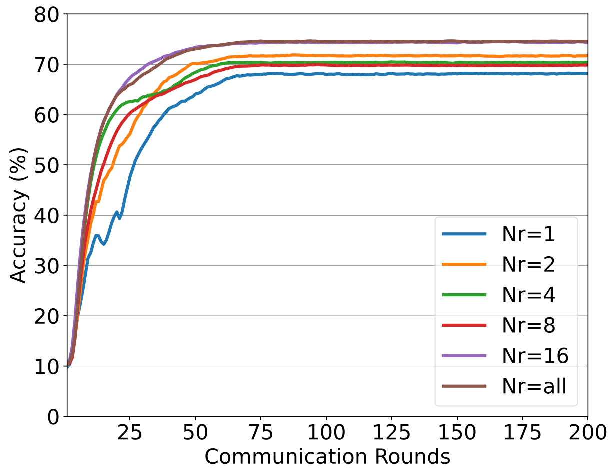


Fig. 8. Impact of N_r in Non-IID settings on Fashion-MNIST.

VI. CONCLUSION

In this article, we introduced a practical problem in the edge computing environment, *Distributed Semi-Supervised Learning*, where each device learns a model collaboratively across the network with mixed private labeled and unlabeled data. To tackle this problem, we proposed a novel method, namely *Distributed Matching* (DistMatch). In DistMatch, each device evaluates all received models from neighboring devices to build a *confidence table*, and generates pseudo-labels according to the confidence table with the help of received models. To avoid self-training with wrong pseudo-labels, each device trains its local model with the *Consensus Consistency* Loss which forces the output of

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

the trained local model to be consistent with pseudo-labels generated by helper models with high consensus. Experiments show that our method outperforms all baselines on several commonly used image classification benchmark datasets. In the future, we plan to extend our methods to solve a more challenging problem in the scenario where some devices in the network do not have any labeled data.

References

[1] P. Bachman, O. Alsharif, and D. Precup, "Learning with pseudo-ensembles," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2014, pp. 3365–3373.

[2] M. Sajjadi, M. Javanmardi, and T. Tasdizen, "Regularization with stochastic transformations and perturbations for deep semi-supervised learning," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2016, pp. 1163–1171.

[3] K. Sohn et al., "FixMatch: Simplifying semi-supervised learning with consistency and confidence," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2020, pp. 596–608.

[4] Y. Xu et al., "Dash: Semi-supervised learning with dynamic thresholding," in *Proc. 38th Int. Conf. Mach. Learn.*, 2021, pp. 11525–11536.

[5] B. Zhang et al., "FlexMatch: Boosting semi-supervised learning with curriculum pseudo labeling," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2021, pp. 18408–18419.

[6] D.-H. Lee et al., "Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks," *Proc. Workshop Challenges Representation Learn.*, vol. 3, 2013.

[7] C. Wei, K. Shen, Y. Chen, and T. Ma, "Theoretical analysis of self-training with deep networks on unlabeled data," in *Proc. Int. Conf. Learn. Representations*, 2021.

[8] S. Zhang et al., "How unlabeled data improve generalization in self-training? A one-hidden-layer theoretical analysis," in *Proc. Int. Conf. Learn. Representations*, 2022.

[9] W. Jeong, J. Yoon, E. Yang, and Sung Ju Hwang, "Federated semi-supervised learning with inter-client consistency & disjoint learning," in *Int. Conf. Learn. Representations*, 2021.

[10] B. McMahan et al., "Communication-efficient learning of deep networks from decentralized data," *Proc. 20th Int. Conf. Artif. Intell. Statist.*, vol. 54, pp. 1273–1282, 2017.

[11] T. Li et al., "Federated optimization in heterogeneous networks," in *Proc. Mach. Learn. Syst.*, vol. 2, pp. 429–450, 2020.

[12] Y. Zhao et al., "Federated learning with non-IID data," 2018, *arXiv: 1806.00582*.

[13] Z. Jiang, A. Balu, C. Hegde, and S. Sarkar, "Collaborative deep learning in fixed topology networks," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2017, pp. 5904–5914.

[14] H. Tang et al., "D^2: Decentralized training over decentralized data," in *Proc. 35th Int. Conf. Mach. Learn.*, 2018, pp. 4848–4856.

[15] R. Ormándi, I. Hegedűs, and M. Jelasity, "Gossip learning with linear models on fully distributed data," *Concurrency Comput. Pract. Exper.*, vol. 25, no. 4, pp. 556–571, 2013.

[16] I. Hegedűs, G. Danner, and M. Jelasity, "Gossip learning as a decentralized alternative to federated learning," in *Proc. 19th IFIP Int. Conf. Distrib. Appl. Interoperable Syst.*, Cham, 2019, pp. 74–90.

[17] S. Itahara et al., "Distillation-based semi-supervised federated learning for communication-efficient collaborative training with non-IID private data," *IEEE Trans. Mob. Comput.*, vol. 22, no. 1, pp. 191–205, Jan. 2023.

[18] I. Bistritz, A. Mann, and N. Bambos, "Distributed distillation for on-device learning," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2020, pp. 22593–22604.

[19] A. Albaseer, B. S. Ciftler, M. M. Abdallah, and A. I. Al-Fuqaha, "Exploiting unlabeled data in smart cities using federated edge learning," in *Proc. 16th Int. Wirel. Commun. Mob. Comput.*, 2020, pp. 1666–1671.

[20] E. Diao, J. Ding, and V. Tarokh, "SemiFL: Semi-supervised federated learning for unlabeled clients with alternate training," *Proc. Adv. Neural Inf. Proces. Syst.*, vol. 35, pp. 17871–17884, 2022.

[21] L. Che et al., "FedTriNet: A pseudo labeling method with three players for federated semi-supervised learning," in *Proc. IEEE Int. Conf. Big Data*, 2021, pp. 715–724.

[22] Z. Zhang et al., "Improving semi-supervised federated learning by reducing the gradient diversity of models," in *Proc. IEEE Int. Conf. Big Data*, 2021, pp. 1214–1225.

[23] C. Fan, J. Hu, and J. Huang, "Private semi-supervised federated learning," in *Proc. 31st Int. Joint Conf. Artif. Intell.*, 2022, pp. 2009–2015.

[24] X. Liang, Y. Lin, H. Fu, L. Zhu, and X. Li, "RSCFed: Random sampling consensus federated semi-supervised learning," in *Proc. IEEE Comput. Soc. Conf. Comput. Vis. Pattern. Recognit.*, 2022, pp. 10144–10153.

[25] C. Li, X. Li, and J. Ouyang, "Semi-supervised text classification with balanced deep representation distributions," in *Proc. Annu. Meet. Assoc. Comput. Linguist. Int. Jt. Conf. Nat. Lang. Process.*, 2021, pp. 5044–5053.

[26] D. Chen, W. Wang, W. Gao, and Zhi-Hua Zhou, "Tri-net for semi-supervised deep learning," in *Proc. 27th Int. Joint Conf. Artif. Intell.*, 2018, pp. 2014–2020.

[27] Y. Grandvalet and Y. Bengio, "Semi-supervised learning by entropy minimization," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2004, pp. 529–536.

[28] S. Laine and T. Aila, "Temporal ensembling for semi-supervised learning," in *Proc. Int. Conf. Learn. Representations*, 2017.

[29] A. Tarvainen and H. Valpola, "Mean teachers are better role models: Weight-averaged consistency targets improve semi-supervised deep learning results," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2017, pp. 1195–1204.

[30] D. Berthelot et al., "MixMatch: A holistic approach to semi-supervised learning," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2019, pp. 5050–5060.

[31] Q. Xie et al., "Unsupervised data augmentation for consistency training," *Proc. Adv. Neural Inf. Proces. Syst.*, vol. 33, pp. 6256–6268, 2020.

[32] D. Berthelot et al., "ReMixMatch: Semi-supervised learning with distribution matching and augmentation anchoring," in *Proc. Int. Conf. Learn. Representations*, 2020.

[33] Y. Jin, X. Wei, Y. Liu, and Q. Yang, Towards utilizing unlabeled data in federated learning: A survey and prospective," 2020, *arXiv:2002.11545*.

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

[35] C. He, M. Annavaram, and S. Avestimehr, "Group knowledge transfer: Federated learning of large CNNs at the edge," *Proc. Adv. Neural Inf. Proces. Syst.*, vol. 33, pp. 14068–14080, 2020.

[36] T.-C. Chiu, Y.-Y. Shih, A.-C. Pang, C.-S. Wang, W. Weng, and C.-T. Chou, "Semisupervised distributed learning with non-IID data for AIoT service platform," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9266–9277, Oct. 2020.

[37] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Comput. Soc. Conf. Comput. Vision. Pattern. Recognit.*, 2016, pp. 770–778.

[38] E. D. Cubuk, B. J. ZophShlens, and Q. Le, "RandAugment: Practical automated data augmentation with a reduced search space," in *Proc. Adv. Neural Inf. Proces. Syst.*, 2020, pp. 18613–18624.

[39] R. T. McDonald, K. B. Hall, and G. Mann, "Distributed training strategies for the structured perceptron," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics: Hum. Lang. Technol.*, 2010, pp. 456–464.

[40] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-MNIST: A novel image dataset for benchmarking machine learning algorithms," 2017, *arXiv: 1708.07747*.

[41] A. Krizhevsky and G. Hinton, "Learning multiple layers of features from tiny images," 2009. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[42] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, and A. Y. Ng, "Reading digits in natural images with unsupervised feature learning," 2011. [Online]. Available: http://ufldl.stanford.edu/housenumbers/nips2011_housenumbers.pdf

[43] A. Coates, A. Y. Ng, and H. Lee, "An analysis of single-layer networks in unsupervised feature learning," in *Proc. Int. Conf. Artif. Intell. Stat.*, 2011, pp. 215–223.

[44] L. Yang, F. Wen, J. Cao, and Z. Wang, "EdgeTB: A hybrid testbed for distributed machine learning at the edge with high fidelity," *IEEE Trans. Parallel Distrib. Syst.*, vol. 33, no. 10, pp. 2540–2553, 2022.

[45] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proc. IEEE*, vol. 86, no. 11, pp. 2278–2324, Nov. 1998.

Hao-Rui Chen received the BE and ME degrees in 2020 and 2023, respectively, from the South China University of Technology, China, where he is currently working toward the doctor's degree with the School of Software Engineering. His research interests include edge computing and edge machine learning.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.

Lei Yang received the BS degree in electronic engineering from Wuhan University, in 2007, the MS degree in computer science from the Institute of Computing Technology, Chinese Academy of Sciences, in 2010, and the PhD degree in computer science from Hong Kong Polytechnic University, in 2014. He is currently a professor with the School of Software Engineering, South China University of Technology (SCUT). Before he joined SCUT in 2016, he was working as a postdoctoral fellow with the Department of Computing, The Hong Kong Polytechnic University. His research interests include cloud and edge computing, distributed machine learning, and federated learning. He has published more than 50 papers in major international journals and conference proceedings including top journals like *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Computers*, *IEEE Transactions on Parallel and Distributed Systems*, *IEEE Transactions on Services Computing*, *IEEE Transactions on Cloud Computing*, *ACM Transactions on Knowledge Discovery from Data*, and top conferences like IEEE INFOCOM and IEEE PERCOM. He also directed several research and development projects from both government and industry agencies.

Xinglin Zhang (Member, IEEE) received the BE degree from the School of Software from Sun Yat-sen University, in 2010, and the PhD degree from the Department of Computer Science and Engineering, Hong Kong University of Science and Technology, in 2014. He is currently a professor with the South China University of Technology. His research interests include mobile crowdsensing, edge computing, and federated learning. He is a member of the the ACM.

Jiaxing Shen received the BE degree in software engineering from Jilin University, in 2014, and the PhD degree in computer science from the Hong Kong Polytechnic University, in 2019. He is an assistant professor with the Department of Computing and Decision Sciences, Lingnan University. He was a visiting scholar with the Media Lab, Massachusetts Institute of Technology, in 2017. His research theme is human dynamics, which aims to understand human behavior and provide actionable insights via cross-disciplinary approaches. Under the theme, his research interests include mobile computing, data mining, and IoT systems. His research has been published in top-tier journals such as *IEEE Transactions on Mobile Computing*, *IEEE Transactions on Knowledge and Data Engineering*, *ACM Transactions on Information Systems*, and ACM IMWUT. He was awarded conference best paper twice including one from IEEE INFOCOM 2020.

Jiannong Cao (Fellow, IEEE) received the BSc degree in computer science from Nanjing University, China, in 1982, and the MSc and PhD degrees in computer science from Washington State University, USA, in 1986 and 1990, respectively. He is currently the Otto Poon charitable foundation professor in data science and a chair professor with the Department of Computing, Hong Kong Polytechnic University, Hong Kong. His research interests include distributed systems and blockchain, wireless sensing and networking, Big Data and machine learning, and mobile cloud and edge computing. He has published five co-authored, nine co-edited books, and more than 500 papers in major international journals and conference proceedings. He is a member of Academia Europaea and ACM distinguished member.

Authorized licensed use limited to: Hong Kong Polytechnic University. Downloaded on May 27,2024 at 10:32:59 UTC from IEEE Xplore. Restrictions apply.