PPFM: An Adaptive and Hierarchical Peer-to-Peer Federated Meta-Learning Framework

Zhengxin Yu, Yang Lu, Plamen Angelov, Neeraj Suri
School of Computing and Communications, Lancaster University, United Kingdom
Email: {z.yu8, y.lu44, p.angelov, neeraj.suri} @lancaster.ac.uk

***Abstract*—With the advancement in Machine Learning (ML) techniques, a wide range of applications that leverage ML have emerged across research, industry, and society to improve application performance. However, existing ML schemes used within such applications struggle to attain high model accuracy due to the heterogeneous and distributed nature of their generated data, resulting in reduced model performance. In this paper we address this challenge by proposing PPFM: an adaptive and hierarchical Peer-to-Peer Federated Meta-learning framework. Instead of leveraging a conventional static ML scheme, PPFM uses multiple learning loops to dynamically self-adapt its own architecture to improve its training effectiveness for different generated data characteristics. Such an approach also allows for PPFM to remove reliance on a fixed centralized server in a distributed environment by utilizing peer-to-peer Federated Learning (FL) framework. Our results demonstrate PPFM provides significant improvement to model accuracy across multiple datasets when compared to contemporary ML approaches.**

***Index Terms*—Meta-learning, federated learning, deep learning, edge computing**

I. Introduction

A growing number of emerging applications (social media, mobile transactions, autonomous vehicles, navigation, etc) are conducted on distributed architectures, whereby distributed users request and/or generate data from disparate edge nodes. As Machine Learning (ML) techniques are increasingly leveraged to analyze such datasets, the application of distributed model training over edge nodes is gaining traction [1] in order to improve application performance and user experience [2].

ML models are typically constructed using a centralized server that collates data with similar patterns to extract common information for classification, prediction, and anomaly detection [3]. However, datasets generated by distributed edge nodes exhibit differing data characteristics with limited datasize [4] and heterogeneous non-iid patterns [5] that both reduce ML model accuracy. Furthermore, centralized data gathering within a distributed environment raises privacy concerns to individual edge nodes, as in certain scenarios it is impractical to establish a centralized server to conduct data gathering and model training [6]. Federated learning (FL) has been considered as a promising ML approach to protect users' privacy, since multiple users collaboratively train a global model without sharing their local data [7]. However, typical FL leverages a common global model for all the users and operates using a fixed centralized server potentially unsuitable to edge nodes with non-iid data within a distributed environment. As distributed environments expand to match distributed applications, there is need for new FL approaches capable of training accurate ML models over limited-size, heterogeneous datasets whilst preserving edge node privacy without reliance upon a fixed centralized server.

We propose PPFM (Peer-to-Peer Federated Meta-learning) to utilize and extend meta-learning and peer-to-peer Federated Learning approaches to adapt the ML architecture to characteristics of varied distributed datasets. PPFM provides a defragmented ML approach whereby discrete data clusters are dynamically generated and associated to a best suited ML schema that matches data cluster characteristics. PPFM leverages three levels: *task level* performs task-specific model training of edge nodes within data clusters exhibitng similar data patterns. *Cluster level* trains cluster-level meta models via a peer-to-peer FL framework, with local models per data cluster gathered to extract common task knowledge. *Global level* adapts the peer-to-peer FL framework to train a global-level meta model, where cluster-level meta models are gathered to extract common cluster knowledge. PPFM performs asynchronous training across levels to speed-up training, and avoids reliance on a fixed centralized server by selecting a new centralized server per model update round when performing cluster level and global level meta-modelling.

To enable fast adaptation, within each cluster a weighted aggregation of associated cluster-level and global-level meta models are in turn fed back to the edge nodes to provide an initial training model for local model updates. In particular, for clusters with small-size data, a large weight can be assigned to the global-level meta model to compensate for training data deficiencies (*e.g.* data size). In contrast, for a cluster with adequate data, a large weight can be assigned to the associated cluster-level meta model in order to place more emphases on the cluster's own data patterns.

Our main contributions in this paper are as follows:

- A novel hierarchical meta-learning architecture is proposed to adaptively match the characteristics of limited-size and heterogeneous data in a distributed environment to improve ML accuracy and efficiency.
- A peer-to-peer FL framework is designed to enhance privacy of individual edge nodes and remove reliance on a fixed centralized server in a distributed environment.
- Extensive experiments with real-world datasets to demonstrate superior accuracy of our method to handle heterogeneous data compared with current ML approaches.

Section II reviews related work. Section III outlines PPFM system architecture and algorithm design. PPFM techniques are detailed in Section IV. Section V presents the evaluation of PPFM. Section VI concludes this paper.

II. Related Works

Meta-learning performs efficient model adaptation for new ML tasks with limited-size training data [8]. Existing approaches can be categorized into: *Optimization-based* approaches treat the training process as an optimization problem. Finn *et al.* [9] proposed optimization-based meta-learning to learn model initialization, so that the model can efficiently adapt to unseen tasks within few gradient descent steps. *Model-based* approaches first train a neural network model to recognize tasks patterns with sampled trajectories, and adjust trained model outputs based on new environmental states to enforce fast learning for unseen tasks. Nagabandi *et al.* [10] proposed an online adaptation mechanism to achieve sample-efficient model-based reinforcement learning. *Metric-based* approaches learn feature representation by employing non-parametric learning techniques. Chen *et al.* [11] developed a generic variational metric scaling framework. However, these meta-learning approaches only consider ML tasks with similar/homogeneous data patterns, and provide limited training and performance adaptation for heterogeneous patterns. Moreover, works are designed for centralized settings, and subject to privacy concerns when applied to distributed environments.

Federated learning (FL) [7] provides a framework to enhance privacy of individual edge nodes when applying meta-learning to distributed environments. FL is a collaborative ML framework, where only local models are shared to conduct global model aggregation in the learning process, while training data of individual edge nodes are never released. Many FL approaches have been developed [7], including secure aggregation [12], compression [13], resource allocation [14], etc. Wang *et al.* [3] analyzed convergence bound for FL with non-iid data distributions. To prevent extraction attacks and collusion threats, Truex *et al.* [12] developed privacy-preserving FL by combining secure multiparty computation with differential privacy. Sattler *et al.* [13] developed a communication-efficient FL framework by compressing both upstream and downstream communications to non-iid data distributions. Zhao *et al.* [14] proposed an FL optimization scheme by considering dynamic wireless channels and different user computational capability.

Federated Meta-learning: Recent works have explored integrating meta-learning with FL to achieve fast learning with limited-size data and protecting individual users' privacy. Jiang *et al.* [15] interpreted an FL algorithm as a meta-learning algorithm to train personalization models. Similar to [15], Fallah *et al.* [16] developed a personalized variant of an FL algorithm by finding an initial shared model to easily adapt to the users' own local datasets. Lin *et al.* [17] introduced a platform-aided collaborative learning framework to achieve real-time edge intelligence at target edge nodes by using a federated meta-learning approach. Existing works do not consider heterogeneous data patterns, and require a fixed centralized server to conduct model training and adaptation, which may not operate effectively in a distributed environment.

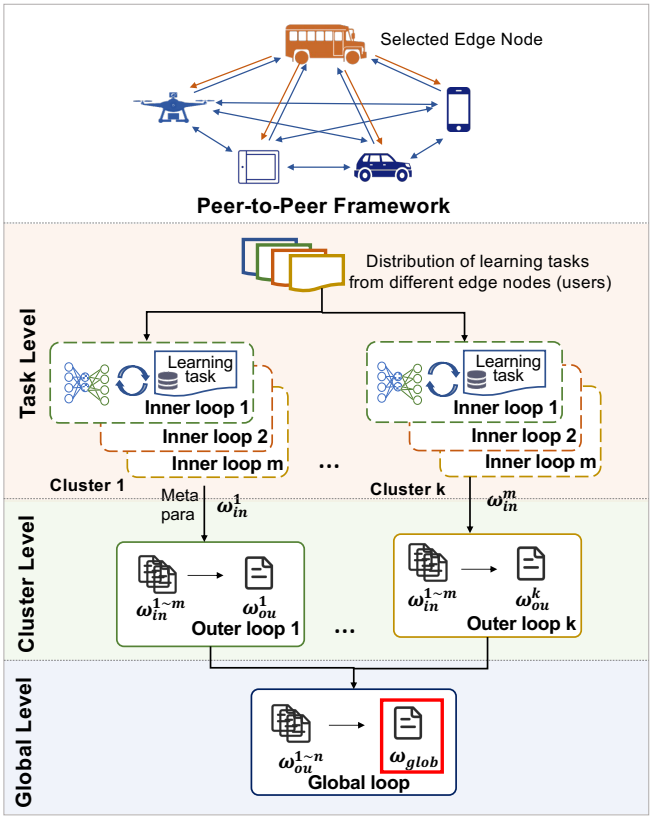


Fig. 1. System architecture of PPFM

III. PPFM System Architecture

A. Distributed Environment Scenario

We consider a distributed network consisting of multiple edge nodes, such as vehicles, sensors, and smart devices, that communicate with each other via wireless links. The edge nodes are moving and can join and leave the network in real time. Each edge node has a learning task with a set of local training data. Due to constraints on, *e.g.*, real-time data gathering and storage capability, the size of an individual edge node's training data is usually limited [18]. In addition, different edge nodes may have significantly different learning tasks and their training data could possess highly heterogeneous patterns/distributions [19]. It is difficult to use the same ML model to effectively process such different data distributions. To address this issue, we aim to match dedicated ML schemas to enable the edge nodes to achieve efficient ML for their tasks over the distributed network. On this premise, the following subsection develops our proposed solution.

B. PPFM Components

The proposed PPFM integrates two components, namely, (1) a defragmented meta-learning framework for achieving accurate and efficient ML over limited-size and heterogeneous data, and (2) a peer-to-peer FL framework for provisioning

the privacy of individual edge nodes as well as removing the reliance on a fixed centralized server. As shown in Fig. 1, the defragmented meta-learning framework consists of three levels: task level, cluster level, and global level. The task level focuses on task-specific model training for the individual tasks of the edge nodes linked to that task. We term this as executing the "inner loops". The cluster level aims to extract their common knowledge (termed as "outer loops"), while the global level aims to further extract global common knowledge covering the tasks of all the edge nodes distributed in different clusters through a global loop. In particular, the meta model extractions at the cluster level and the global level are carried out by the proposed peer-to-peer FL framework.

The overall design of PPFM is summarized by Algorithm 1 and informally elaborated next with detailed analysis in Section 3. As outlined in Algorithm 1, the proposed PPFM has four functional blocks: clustering, task-level training, cluster-level meta training, and global-level meta training.

Clustering. We assume PPFM is running on N edge nodes. These edge nodes are first partitioned into K clusters, where each cluster includes a subset of edge nodes whose training datasets share similar patterns. In particular, from line 2 to 6, each edge node i utilizes Variational AutoEncoder (VAE) to extract a feature vector z_i to represent its data D_i. Then, from line 7 to 8, the K clusters are generated by using a K-means algorithm based on the learned feature vectors, where the metric Kullback-Leibler (KL) divergence is used to calculate the distance between feature vectors. The details of feature extraction and cluster generation are provided in Section IV-A.

Task-level training. In the task level, each edge node i is assigned to its respective cluster in order to learn a local task-level ML model w_{in}^{i}. To do this, within each cluster k, at the first task-level communication round, each edge node i first arbitrarily initializes its local model. After the first communication round, the initial local model is a weighted aggregation of the cluster-level meta model $w_{ou}^{k,r}$ of cluster k and the global-level meta model w_{glob}^{g}. Note that r is the number of cluster-level communication rounds for the update of outer loops and g is the number of global-level communication rounds for the update of global loops. The details are presented in Eq. 4 in Section IV-B. The weights are proportional to the data size of the edge node i. This initialization is essential for achieving fast adaptation by taking into account both cases of limited-size data and heterogeneous patterns. After the initialization, from line 18-23, each edge node i updates its local model $w_{in}^{i,r}$ over its local training data D_i by using $LocalUpdate$ in the r^{th} communication round. The details of local model training $LocalUpdate$, are provided in Section IV-B and Algorithm 2. The updated local model $w_{in}^{i,r}$ will be sent to the cluster master node N_{clu}^{k}.

Cluster-level meta training. For each cluster k, line 24 iteratively develops and refines a cluster-level meta model $w_{ou}^{k,r}$, which aims to extract the common features from all task-level models. The details for model extraction are provided in Section IV-B. The updated meta model will be uploaded to the global master node N_{glob}^{g}.

Algorithm 1: PPFM runs on N edge nodes (indexed by i). Each node has a local dataset $D_i = \{x_i\}_{j=1}^{X}$.

1 **Procedure** DataClustering
2 **VAE Feature Extraction**
3 **for** each node $i \in \{1, 2, \cdots, N\}$ **in parallel do**
4 Parameters of VAE are updated by using SGD based on Eq. (1)
5 $\boldsymbol{z}_i = \mu + \sigma \odot \epsilon$ ▷ Details are described in Sec. IV-A
6 **end**
7 **KL based K-means Clustering**
8 K clusters ← Computed by K-means, based on KL divergence distance (defined in Eq. (2))
9 $\boldsymbol{w}$ ← HierFedMetaLearning(K, i, D_i)
10 **Procedure** HierFedMetaLearning (K, i, D_i)
11 **Global-level Meta Training**
12 **for** each global-level communication round $g \in \{1, \cdots, G\}$ **do**
13 Select one node from all nodes randomly as a global master node, denoted by N_{glo}^{g}
14 **Cluster-level Meta Training**
15 **for** each cluster $k \in \{1, \cdots, K\}$ **in parallel do**
16 Select one node from M nodes randomly as a cluster master node, denoted by N_{clu}^{k}
17 **for** each cluster-level communication round $r \in \{1, \cdots, R\}$ **do**
18 **Task-level Model Training**
19 **for** each edge node $i \in \{1, \cdots, M\}$ **in parallel do**
20 ▷ *LocalUpdate is detailed in Alg. 2*
21 $\boldsymbol{w}_{in}^{i,r+1} \leftarrow \text{LocalUpdate}(\boldsymbol{w}_{in}^{i,r}, k, E)$
22 Send $\boldsymbol{w}_{in}^{i,r+1}$ to N_{clu}^{k}
23 **end**
24 $\boldsymbol{w}_{ou}^{k,r+1}$ is updated by using Eq. (10)
25 **end**
26 Send $\boldsymbol{w}_{ou}^{k,R}$ to N_{glo}^{g}
27 **end**
28 $\boldsymbol{w}_{glob}^{g+1} \leftarrow \frac{1}{K}\sum_{k=1}^{K} w_{\text{ou}}^{k,R}$
29 **Send** $\boldsymbol{w}_{glob}^{g+1}$ to each cluster k
30 **end**
31 **Return** $\boldsymbol{w} \leftarrow \boldsymbol{w}_{glob}^{G}$

Global-level meta training. In the global level, line 28 further constructs a global-level meta model w_{glob} by extracting common features from different cluster-level meta models.

In the above, the functional blocks of clustering, cluster-level meta training, and global meta training require collaborations between the edge nodes, and are carried out by a peer-to-peer FL framework, described next. The number of hierarchy levels within PPFM can be adjusted based on application data characteristics, capable of supporting a two-level, three-level, and higher. The three-level hierarchy as shown in Fig. 1 is a

visual depiction of one instance.

Peer-to-peer FL. Given a set of participating edge nodes, each edge node has a local model. In general, the proposed peer-to-peer FL framework has the following three steps. (i) Master node selection: the base station connected by edge nodes is responsible for randomly selecting a master node for each cluster. (ii) Local model update and gathering: each participating edge node updates its local model and sends it to the master node. (iii) Joint model update: the master node performs required operations over the local models of the participating edge nodes and sends required updated outputs to the other participating edge nodes.

IV. PPFM Techniques

In this section, we elaborate the details of our proposed PPFM. We first describe hierarchical tasks clustering based on heterogeneous datasets. Then, we introduce how inner, outer and global loops can collaboratively train ML models to enhance model training efficiency. An asynchronous weighted aggregation algorithm for peer-to-peer federated learning and a weighted model initialization approach are presented.

A. Data Representation Clustering

As shown in Fig. 2, we utilize data features from edge node for clustering. To learn highly effective data representation from the local dataset at each edge, Variational AutoEncoder (VAE) is exploited [20]. With the derived representation vectors, a Kullback-Leibler (KL) divergence based K-means algorithm is further used to cluster the heterogeneous datasets.

We assume each edge node/task has a local dataset $D_i = \{x_i\}_{j=1}^{X}$, consisting of X non-iid samples of some continuous or discrete variable x. VAE is an unsupervised artificial neural network, aiming to copy its input x to its output by learning latent representations z [21]. Typically, a VAE consists of an encoder and decoder. The encoder takes x as input and outputs a distribution $p_\theta(z \mid x)$ of latent representation z, where θ are parameters of a neural network. For the decoder, its input is the representation z and it outputs the parameters to the probability distribution of the data, defined as $q_\phi(x \mid z)$, where ϕ are the parameters of another neural network. The loss function of VAE [21] is defined by

$$\begin{aligned}\ell(\theta, \phi; x_i) = & -\mathbb{E}_{z\sim p_\theta(z|x_i)}\left[\log q_\phi(x_i \mid z)\right] \\ & + \mathbb{KL}\left(p_\theta(z \mid x_i) \parallel p(z)\right).\end{aligned} \tag{1}$$

The first term is the reconstruction loss. We take the expectation of the encoder's distribution over the representations. The reconstruction log-likelihood $\log q_\phi(x_i \mid z)$ is used to measure how effectively the decoder learns to reconstruct an input x given its latent representation z. The second regularization term is the KL divergence [22] between the encoder's distribution $p_\theta(z \mid x)$ and $p(z)$, where $p(z)$ is a prior distribution over latent variables z. This loss term penalizes the VAE if it starts to produce latent vectors that are not from the desired distribution. In order to optimize this loss function of VAE, we use a reparameterization trick to do optimization. $p_\theta(z|x)$ is a Gaussian distribution. The reparameterization trick rewrites the representation of $z \sim p_\theta(z|x) = \mathcal{N}(\mu, \sigma^2)$ into $z = \mu + \sigma \odot \epsilon$, $\epsilon \sim \mathcal{N}(0, 1)$ [21], where μ and σ are a mean and a variance of Gaussian distribution. ϵ is an auxiliary noise variable. VAE can effectively cluster similar input data in the latent space, so we use the output of the encoder network as the representation of edge node to cluster edge nodes.

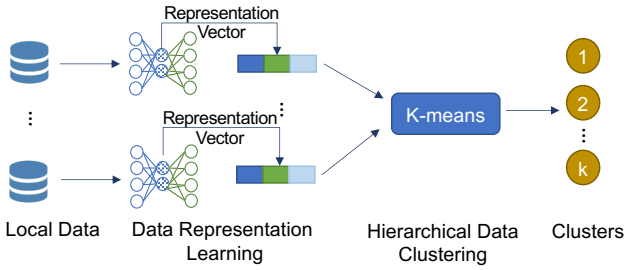


Fig. 2. PPFM data clustering method

Since the given representation of each dataset z_i are sampled from distributions, KL divergence can conveniently be used to measure the difference between these distributions. Thus, a KL divergence based K-means clustering algorithm [23] is utilized to cluster edge nodes. The main steps of the clustering algorithm are given as follows. The first step is to randomly select K representation vectors from the edge nodes as initial cluster centers. The second step is to compute the distance of each calculated representation vector to the cluster centres by using KL divergence that can be defined as

$$\begin{aligned}D_{KL}\left(p_j(z) \parallel p_i(z)\right) &= \sum_z p_j(z) \log \frac{p_j(z)}{p_i(z)} \\ &= \sum_z p_j(z) \log p_j(z) - \sum_z p_j(z) \log p_i(z).\end{aligned} \tag{2}$$

$p_j(z)$ and $p_i(z)$ represent calculated probability distributions where representation vectors z of edge nodes and cluster centers are sampled from, respectively. In the third step, each representation vector is allocated to the closest cluster. The fourth step is that the centre of each cluster is recomputed by taking the mean of representation vectors of edge nodes in a cluster. Steps two to four are repeated until convergence [24]. Additionally, in the distributed environment, new edge nodes may keep arriving continuously. Thus, the new edge node will join K-means during the training process.

B. Hierarchical Federated Meta-Learning

Based on the above data clustering, our goal is to learn hierarchical meta models with strong adaptation ability for tasks within and across clusters. PPFM achieves this goal by jointly optimizing adaption performance to similar tasks within a cluster and the generalizability across heterogeneous clusters of learning tasks. The learning task of edge node i is denoted by T_i where the dataset of the learning task is D_i. Let us denote a set of learning tasks as $\mathcal{T} = \{T_1, T_2, ..., T_n\}$. In PPFM, the learning tasks are divided into K clusters $\widehat{\mathcal{T}} = \{\mathcal{T}^1, \mathcal{T}^2, ..., \mathcal{T}^K\}$ where each cluster of tasks $\mathcal{T}^k$ includes several similar learning tasks $\mathcal{T}^k = \{T_1^k, T_2^k, ..., T_m^k\}$. We

Algorithm 2: Local update in PPFM. M edge nodes are indexed by i; b is batch size; δ is the data weight

Procedure LocalUpdate($\boldsymbol{w}_{in}^{i,r}, k, E$)
  Download $\boldsymbol{w}_{ou}^{r}$ from N_{clu}^{k} and $\boldsymbol{w}_{glob}^{g}$ from N_{glo}^{g}
  $\boldsymbol{w}_{in^*}^{r+1} \leftarrow (1-\delta)\boldsymbol{w}_{glob}^{g} + \delta \boldsymbol{w}_{ou}^{r}$, defined in Eq. (4)
  for each local epoch e from 1 to E **do**
    for mini-batch b **do**
      $\boldsymbol{w}_{in}^{i,r+1} \leftarrow \boldsymbol{w}_{\text{in}^*}^{i,r+1} - \alpha \nabla \mathcal{L}_{T_i^k}\left(w_{\text{in}}^{i,r}, D_i\right) +$
      $\frac{\lambda}{2}\left\|w_{\text{in}}^{i,r} - w_{\text{in}^*}^{i,r+1}\right\|^2$, defined in Eq. (9)
    end
  end
return $\boldsymbol{w}_{in}^{i,r}$ to cluster master node N_{clu}^{k}

assume that there are m nodes in cluster k. The loss function of a learning task T_i^k is defined as $\mathcal{L}_{T_i^k}$. Define $w_{in}^{i,r}$, $w_{\text{ou}}^{k,r}$, and w_{glob}^{g} as the task-specific model parameters of task T_i^k in the inner loop, the meta parameters of the outer loop for cluster $\mathcal{T}^k$, and meta parameters of the global loop, respectively.

The inner loop training in the task level aims to quickly learn an effective task specific model based on the local dataset and the meta parameters of the outer loop so that the training loss is minimal. The inner loop target function is to minimize $\mathcal{L}_{\mathcal{T}^k}(w_{\text{in}}^{i})$. PPMF obtains w_{in}^{i} by optimizing this loss through stochastic gradient descent (SGD), that is:

$$w_{in}^{i,r+1} \leftarrow w_{\text{ou}}^{k,r} - \alpha \nabla \mathcal{L}_{T_i^k}(w_{\text{ou}}^{k,r}), \tag{3}$$

where $w_{\text{ou}}^{k,r}$ represents the calculated meta parameters of outer loop in the r-th round. The pseudocode of the inner loop training is described in Algorithm 2. Note that, for each learning task in the task cluster $\mathcal{T}^k$, the model parameters are initialized by the meta parameters of the outer loop $\mathcal{T}^k$ in the first round. After the first communication round, considering the different amount of data among edge nodes, each edge node contributes differently to the global model. A larger weight is given to the edge node with a larger dataset. Thus, instead of directly using the computed cluster-level meta model as the initial model, we combine the cluster-level meta model with global-level meta model by using two weights. These weights are decided by the proportion of the data size of cluster to the total data size of all participating edge nodes. Thus, the initial model of edge nodes after the first communication round is as follow:

$$w_{\text{in}^*}^{i,r+1} \leftarrow (1-\delta)\, w_{\text{glob}}^{g} + \delta w_{\text{ou}}^{k,r}, \tag{4}$$

where δ is the data weight. $w_{\text{ou}}^{k,r}$ and $\boldsymbol{w}_{glob}^{g}$ are downloaded from N_{clu}^{k} and N_{glo}^{g}, respectively.

The goal of the outer loop training in a task cluster is to learn the meta parameters $\mathcal{T}^k$ so that the corresponding learning task can fast learn effective task-specific model w_i^k. We define $p(\mathcal{T}^k)$ as learning task distribution in cluster $\mathcal{T}^k$. The goal of the outer loop training in the cluster level can be formally defined as

$$\min_{w_{\text{ou}}^{k}} \mathcal{L}_{\mathcal{T}^k}(w_{\text{ou}}^{k}) = \mathbb{E}_{T_i^k \sim p(\mathcal{T}^k)}\left[\mathcal{L}_{T_i^k}(w_{in}^{k,R})\right]. \tag{5}$$

R is the total number of cluster-level rounds. PPMF initializes the meta parameters of the outer loops, w_{glob}^{g}, by the meta parameters of global loop $\mathcal{T}^k$. Since we optimize the model parameters through SGD, thus the meta parameters of the outer loop can be obtained by

$$w_{\text{ou}}^{k,r+1} \leftarrow w_{\text{glob}}^{g} - \beta \nabla \mathbb{E}_{T_i^k \sim p(\mathcal{T}^k)}\left[\mathcal{L}_{T_i^k}(w_i^{k,r})\right]. \tag{6}$$

Here, $w_i^{k,r}$ can be obtained by Eq. (3) and β is the learning rate of the cluster-level loops.

Similarly, the goal of the global loop is to extract common knowledge among clusters, which can speedup the training of outer loop for each cluster. Denote $p(\widehat{\mathcal{T}})$ as the cluster distribution of learning tasks. The learning target of the global loop is to obtain effective global meta parameters so that the loss function of the outer loops in the cluster level is minimal. Formally, we define the global loop loss function as:

$$\min_{w_{\text{ou}}^{k}} \mathcal{L}_{\widehat{\mathcal{T}}}(w_{\text{glob}}^{g}) = \mathbb{E}_{\mathcal{T}^k \sim p(\widehat{\mathcal{T}})}\left[\mathcal{L}_{\mathcal{T}^k}(w_{\text{ou}}^{k,R})\right]. \tag{7}$$

where w_{ou}^{k} can be obtained by Eq. (6). Therefore, the update rule of the global level can be expressed by

$$w_{\text{glob}}^{g} \leftarrow w_{\text{glob}}^{g} - \eta \nabla \mathbb{E}_{\mathcal{T}^k \sim p(\widehat{\mathcal{T}})}\left[\mathcal{L}_{\mathcal{T}^k}(w_{\text{ou}}^{k,r})\right]. \tag{8}$$

where η is the learning rate of the global loop.

Specifically, the optimization of w_{glob}^{g} and w_{ou}^{k} uses second and third derivatives when backpropagating the meta-gradient in Eq. (6) and Eq. (8), respectively. In FL setting, calculating second and third derivatives requires extra communication and computation power, which might not be suitable for edge devices. Hence, we use the first-order approximation method [25] to remove the second and third derivative terms in Eqs. (6) and (8). According to [25], the loss function of inner loop can be rewritten as:

$$\mathcal{L}_{\mathcal{T}^k}\left(w_{\text{in}}^{i,r+1}\right) = \min_{w_{\text{in}}^{i,r}}\left\{\mathcal{L}_{\mathcal{T}_i^k}\left(w_{\text{in}}^{i,r}\right) + \frac{\lambda}{2}\left\|w_{\text{in}}^{i,r} - w_{\text{in}^*}^{i,r}\right\|^2\right\}. \tag{9}$$

$w_{\text{in}}^{i,r}$ is the task-specific model of edge node i in the r-th communication round. λ is a regularization parameter, which decides the contribution of $w_{\text{in}^*}^{i,r}$ to the task-specific model. Therefore, the update of a model in the outer loop is as follows:

$$w_{\text{ou}}^{k,r+1} = (1-\psi)\, w_{\text{ou}}^{k,r} + \psi \sum_{i=1}^{M} \frac{w_{\text{in}}^{i,R}}{|D_i|}. \tag{10}$$

Note that ψ is an additional parameter for model update. In the global loop, the FederatedAveraging (FedAvg) algorithm [7] is applied. The FedAvg based global meta model updates is as follows: $\frac{1}{K}\sum_{k=1}^{K} w_{\text{ou}}^{k,R}$.

Additionally, in the outer loops and the global loop, a weighted asynchronous aggregation algorithm is executed to aggregate the model. Edge nodes contain different computing and storage capabilities, which causes some edge nodes to lag behind or drop out during the training process. A synchronous aggregation algorithm may result in extremely slow training time. Thus, we utilize an asynchronous aggregation algorithm

to conduct the outer and global loops, without waiting for all participating edge nodes to upload their models.

The relationship between inner loop, outer loop and global loop is as follows. The local data at each edge node is the inputs of inner loops and the outputs of inner loops is meta parameters which are the inputs of outer loops. After the outer loop performs the aggregation, the task-level meta-parameters can be obtained. They are the outputs of the outer loop, which are also the inputs to the global loop. The outputs of the global loop is the cluster-level meta parameters. They will be sent back to outer loops, to start a new communication round. Furthermore, the complexity of PPFM algorithm is $O(N)$, where N is the data input size.

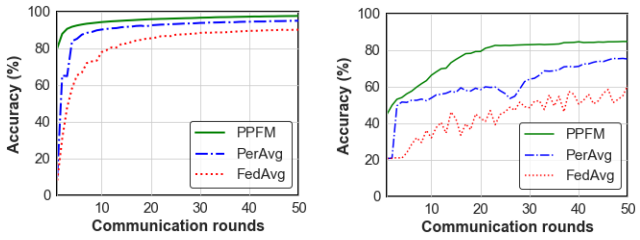
V. Evaluation

In this section, we conduct a variety of experiments to evaluate the performance of PPFM under distributed environments, considering variations of data heterogeneity effects. We compare PPFM with multiple state-of-the-art FL methods to validate the higher accuracy achieved by PPFM.

A. Experiment Settings

*1) **Simulation Environment**:* We utilize a Multi-access Edge Comping (MEC) environment consisting of several vehicles and smart devices. These edge nodes have local datasets on which model training is conducted. The number of edge nodes varies from 5 to 100 similar to contemporary settings [3] [16]. The simulation was conducted by using 3 x HP Z440 workstations with 64G memory. Keras is employed as the ML framework to implement the meta-learning and FL, with TensorFlow as the execution engine.

*2) **Datasets & Models**:* We evaluate the proposed PPFM on four different datasets: two real-world datasets and two synthetic datasets. First, we evaluate the proposed method on classification tasks with standard image datasets: MNIST [26] and CIFAR-10 [27]. MNIST is a database of handwritten digits, containing 60,000 examples in a training set and 10,000 examples in a test set with 10 classes. CIFAR-10 is labeled subsets of 80 million images dataset. It consists of 60000 32x32 color images in 10 classes, with 6000 images per class. Compared with MNIST, this dataset is larger in scale. To evaluate the proposed framework on various heterogeneous data distributions, we generate a synthetic dataset by following a similar setup to the one used in [28]. The parameter γ is used to control the level of heterogeneity. We vary γ to generate four heterogeneous distributed datasets, as shown in Fig. 4. For evaulating PPFM on regression, we utilize the sinusoid function to generate another synthetic dataset [9], where the phase and amplitude of the sinusoid are varied between edge nodes. In our experiments, the phase varies in the range $[1, \pi]$ and the amplitude varies within $[0.1, 5.0]$, and the dimension of input and output is 1. Additionally, three ML models are used to evaluate PPFM, which are Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Conditional Multinomial Logistic Regression (CMLR).



(a) MNIST Dataset (b) CIFAR-10 Dataset

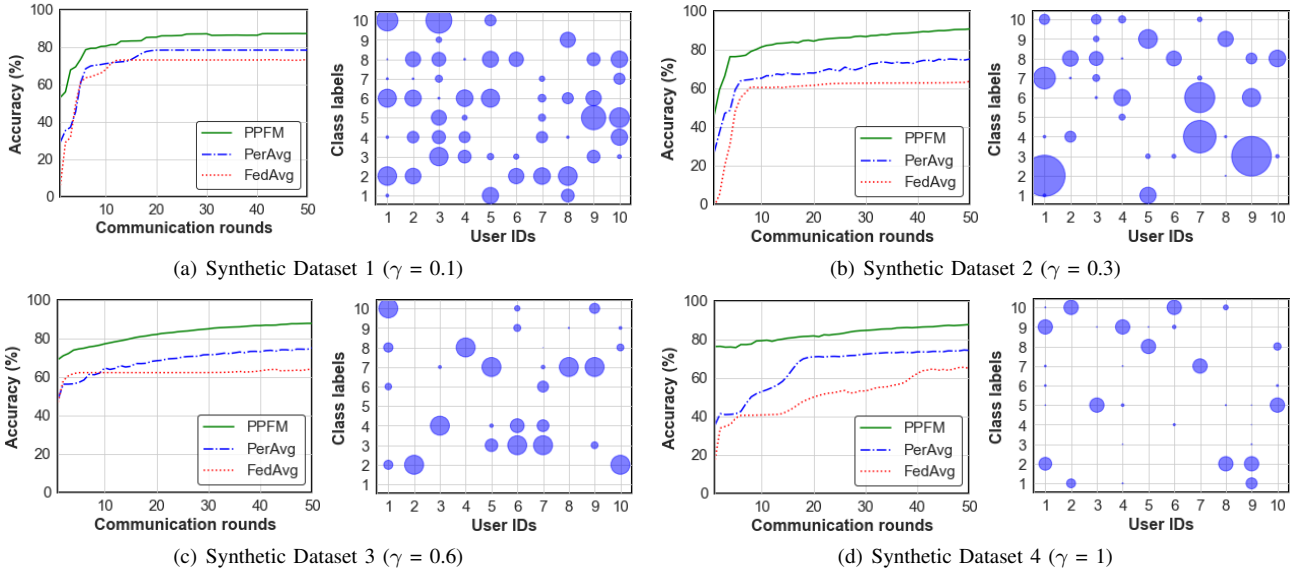
Fig. 3. PPFM vs. Comparative algorithms (Standard Datasets)

*3) **Comparative Algorithms**:* Our proposed framework PPFM is compared with the two FL baseline approaches: Federated Averaging (FedAvg) and Personalized Federated Learning (PerAvg). FedAvg is a baseline of FL that allows local nodes to perform more than one batch update on the local dataset and exchanges the updated weights, instead of the gradients [7]. PerAvg is a state-of-the-art method of FL [16], that combines meta-learning to learn more personalized model for each node. Algorithm configuration values match those within each paper.

B. Performance Evaluation

Evaluation for Classification Tasks: Figs. 3 show the effectiveness of our proposed method PPFM for classification tasks in terms of accuracy by comparing it with FedAvg and PerAvg. With the increase of communication rounds, the accuracy of all algorithms increases. Our proposed PPFM outperforms FedAvg and PerAvg. This is because PPFM utilizes multiple meta-models to extract common knowledge from all edge nodes. PerAvg and FedAvg use one meta-model/model to process the heterogenous data. The accuracy of PerAvg is higher than FedAvg, since it uses meta-learning to train more personalized models for edge nodes. In contrast, FedAvg only conducts a global model for all edge nodes, which results in low model accuracy for individual edge nodes. As shown in Fig. 3 (a), the dataset we used is MNIST and DL model is Deep Neural Network (DNN). The accuracy of PPFM is 95.13% when federated communication rounds are 10, while PerAvg achieves 91.12% and FedAvg is only 78.25%. PPFM can reach optimal accuracy in a few communication rounds. Thus, PPFM can also improve the efficiency of model training. The same trend has been observed in Fig. 3 (b). The dataset of CIFAR-10 and the DL model of Convolutional Neural Network (CNN) are used in this experiment. After 20 rounds, the accuracy of PPFM can achieve 80%; PerAvg gets 60%; FedAvg is 42%. CIFAR-10 dataset is more complex than MNIST, thus the model accuracy is lower than MNIST. 20 edge nodes participate in these two experiments.

Data Distribution Effects: Figs. 4 demonstrate the influence of data distribution on learning performance of the model in PPFM. We use four different levels of heterogeneity non-iid data distribution datasets to evaluate PPFM. The right side of each subfigure in Figs. 4 shows the data distribution of the edge node. Each of them contains various number



(a) Synthetic Dataset 1 (γ = 0.1)

(b) Synthetic Dataset 2 (γ = 0.3)

(c) Synthetic Dataset 3 (γ = 0.6)

(d) Synthetic Dataset 4 (γ = 1)

Fig. 4. PPFM vs. Comparative algorithms (Synthetic Datasets)

TABLE I
COMPARISON OF PPFM AND PPFM WITHOUT HIERARCHICAL ARCHITECTURE

		PPFM		PPFM without hierarchical architecture	
ML Model	Dataset	Accuracy	Time / per round	Accuracy	Time / per round
DNN	non-iid MNIST	95.13%	4.86s	90.68%	1.15s
CNN	non-iid CIFAR-10	83.97%	32.6s	79.73%	17.81s
MCLR	Synthetic	92.46%	3.13s	87.21%	0.92s

of classes and data samples. There are 10 participated edge nodes in these experiments. Data heterogeneity increases from Synthetic Dataset 1 in Fig. 4 (a) to Synthetic Dataset 4 in Fig. 4 (d). Figs. 4 show that PPFM has different advantages for different data distributions. As expected, PPFM achieves the highest accuracy among other reference algorithms, as it utilises hierarchical learning loops to analyze heterogeneous data. More fine-grained data features are extracted, so the accuracy of PPFM can be further improved. Compared to FedAvg, PPFM improves model accuracy by around 20% on average. For instance, as shown in Fig. 4 (a), PPFM achieves 57% in the first communication round, while PerAvg gets 31% and FedAvg is about 7%. After 20 communication rounds, PPFM almost obtains the optimal accuracy, which is about 85%. Meanwhile, PerAvg is 79.5% and FedAvg is 73%. As data heterogeneity increases, PPFM shows significant advantages in model accuracy and training efficiency, as shown in Figs. 4 (b), (c) and (d). It always can train a more accurate model and achieve optimal accuracy faster than PerAvg and FedAvg. When the value of γ is 1, the accuracy of PPFM in the first communication round reaches 78% and it can achieve about 86% after 50 communication rounds. In contrast, PerAvg can only get 38% and FedAvg is 19% in the first round. When the communication round is 50, the accuracy of PerAvg and FedAvg are 77% and 66%, respectively.

Hierarchical Architecture Effects: Table I compares the model accuracy in PPFM and PPFM without hierarchical architecture. In this experiment, three ML models (CNN, DNN, and CMLR) are utilized by using three datasets (MNIST, CIFAR-10, and Synthetic). For non-iid data distribution datasets, the accuracy of PPFM is significantly higher than PPFM without hierarchical architecture, since multiple meta-models collaboratively to deal with heterogeneous data across different edge nodes. For example, the accuracy of the DNN model in three-level PPFM is 95.13% by using non-iid MNIST. In contrast, DNN in PPFM without hierarchical architecture can only achieve 90.68%. The accuracy of CNN and MCLR under the three-level architecture are 83.97% and 92.46%, while without a three-level framework they only obtain 79.73% and 87.21%, respectively.

Additionally, experimental results in Table I present a trade-off between accuracy and training time for the non-iid data distribution. Higher accuracy can be achieved by using more time. In PPFM with three-level architecture, per FL communication round of training DNN takes 4.86s with the non-iid dataset. In contrast, training the same DNN model in PPFM without hierarchical architecture only needs 1.15s. Training time is greatly reduced. Similar trends are obtained for training other models (CNN and MCLR). Thus, there are some limitations of PPFM. PPFM takes more time to

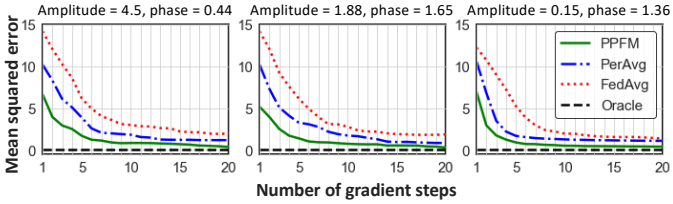


Fig. 5. Sinusoid regression results (Synthetic Dataset)

train a model for per communication rounds. As we can see from Fig. 4 (a), PPFM may achieve similar accuracy with other FL based methods if the dataset is iid or low-level data heterogeneity, but it has high computational cost. Our method is more suitable for heterogeneous data. Furthermore, in our proposed PPFM, we use the cluster level to optimize the meta-model adaptability of similar data distribution, in order to further process non-iid data distribution and accelerate new edge nodes to train their model. However, using clustering algorithms in PPFM is less efficient if all edge nodes have similar data characteristics. In this case, all edge nodes can be assumed in one cluster, therefore, the global level in PPFM is not necessary. It is because the global level is supposed to learn common knowledge from all different clusters.

Evaluation for Regression Tasks: Fig. 5 presents the efficiency of PPFM on regression tasks. Each edge node involves a regression task that aims to fit a sine wave. Data x are sampled uniformly from $[-5.0, 5.0]$. The regressor in this experiment is a multilayer perceptron model with 1 hidden layer of size 10 with ReLU nonlinearity. We evaluate the performance of PPFM by fine-tuning the model that is trained by PPFM on different data distribution. Oracle is the baseline, which presents the best performance. Figs. 5 show the learning curve at test time with varying numbers of test-time samples, amplitude, and phase. After 20 gradient steps, the mean squared error is close to the oracle which is 0. The mean squared error of PPFM reduces with the number of gradient steps rising. PerAvg and FedAvg also show a similar trend, but the mean squared error is higher than PPFM. For example, when amplitude is 4.5 and phase is 0.44, the mean squared error of PPFM is 6 after the first gradient steps, while PerAvg is 10.2 and FedAvg is 14.5. When the gradient step is 20, PPFM decreases to 0.9, PerAvg is 1.4 and FedAvg is 2.2.

VI. Conclusions

In this paper, we have proposed PPFM - a new defragmented federated meta-learning architecture capable of adaptively matching varying data characteristics by dynamically generating learning loops. We have also integrated a peer-to-peer FL into PPFM to effectively operate within dynamic distributed environments. Our experiment results show that PPFM can improve the accuracy and efficiency of ML models for processing heterogeneous data, allowing multiple meta-models to collaboratively extract common features from edge nodes. In our future work, we intend to further build upon PPFM to design an adaptive meta reinforcement learning framework for edge computing, to enable lifelong learning ability in a distributed dynamical environment.

Acknowledgments

Research supported by the UKRI Trustworthy Autonomous Systems Node in Security/EPSRC grant EP/V026763/1.

References

[1] Yujing Chen et al. Asynchronous online federated learning for edge devices with non-iid data. In *Proc. of Big Data*. IEEE, 2020.
[2] Jin-A Choi and Kiho Lim. Identifying machine learning techniques for classification of target advertising. *ICT Express*, 6(3):175–180, 2020.
[3] Shiqiang Wang et al. When edge meets learning:adaptive control for resource-constrained distributed machine learning. In *Proc. of INFOCOM*. IEEE, 2018.
[4] Shen Zhang et al. Few-shot bearing anomaly detection via model-agnostic meta-learning. In *Proc. of ICEMS*. IEEE, 2020.
[5] Yuchang Sun et al. Semi-decentralized federated edge learning for fast convergence on non-iid data. *arXiv preprint arXiv:2104.12678*, 2021.
[6] Latif Khan et al. Federated learning for edge networks: Resource optimization and incentive mechanism. *IEEE Communications Magazine*, 2020.
[7] Brendan McMahan et al. Communication-efficient learning of deep networks from decentralized data. In *Proc. of AISTATS*, 2017.
[8] Timothy Hospedales et al. Meta-learning in neural networks: A survey. *arXiv preprint arXiv:2004.05439*, 2020.
[9] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *Proc. of ICML*, 2017.
[10] Anusha Nagabandi et al. Learning to adapt in dynamic, real-world environments through meta-reinforcement learning. *arXiv preprint arXiv:1803.11347*, 2018.
[11] Jiaxin Chen et al. Variational metric scaling for metric-based meta-learning. In *Proc. of AAAI*, 2020.
[12] Stacey Truex et al. A hybrid approach to privacy-preserving federated learning. In *ACM Workshop on Artificial Intelligence and Security*, 2019.
[13] Felix Sattler et al. Robust and communication-efficient federated learning from non-iid data. *IEEE trans. on neural networks and learning systems*, 2019.
[14] Zichao Zhao et al. System optimization of federated learning networks with a constrained latency. *IEEE Trans. on Vehicular Technology*, 2021.
[15] Yihan Jiang et al. Improving federated learning personalization via model agnostic meta learning. *arXiv preprint arXiv:1909.12488*, 2019.
[16] Alireza Fallah et al. Personalized federated learning: A meta-learning approach. *arXiv preprint arXiv:2002.07948*, 2020.
[17] Sen Lin et al. A collaborative learning framework via federated meta-learning. In *Proc. of ICDCS*. IEEE, 2020.
[18] Zeyi Tao and Qun Li. eSGD: Communication efficient distributed deep learning on the edge. In *USENIX HotEdge*, 2018.
[19] Techuan Chiu et al. Semisupervised distributed learning with non-iid data for aiot service platform. *IEEE Internet of Things Journal*, 2020.
[20] Carl Doersch. Tutorial on variational autoencoders. *arXiv preprint arXiv:1606.05908*, 2016.
[21] Diederik Kingma and Max Welling. Auto-encoding variational bayes. *arXiv preprint arXiv:1312.6114*, 2013.
[22] Stephen Odaibo. Tutorial: Deriving the standard variational autoencoder (vae) loss function. *arXiv preprint arXiv:1907.08956*, 2019.
[23] Iwan Tri Riyadi Yanto et al. A framework of mutual information kullback-leibler divergence based for clustering categorical data. *International Journal on Informatics Visualization*, 2021.
[24] Jingjing Cui et al. Unsupervised machine learning-based user clustering in millimeter-wave-noma systems. *IEEE Trans. on Wireless Communications*, 2018.
[25] Canh T Dinh et al. Personalized federated learning with moreau envelopes. *arXiv preprint arXiv:2006.08848*, 2020.
[26] Yann LeCun et al. MNIST handwritten digit database. 2010.
[27] Alex Krizhevsky and Geoffrey Hinton. Learning multiple layers of features from tiny images. *MSc Thesis, University of Toronto*, pages 1–54, 2009.
[28] Tian Li et al. Federated optimization in heterogeneous networks. *Proc. of MLSys*, 2020.