Blockchain-Empowered Federated Learning Through Model and Feature Calibration

Qianlong Wang, Weixian Liao, Yifan Guo, *Member, IEEE*, Michael McGuire, and Wei Yu, *Senior Member, IEEE*

***Abstract*—With the proliferation of computationally powerful edge devices, edge computing has been widely adopted for wide-ranging computational tasks. Among these, edge artificial intelligence (AI) has become a new trend, allowing local devices to work cooperatively and build deep learning models. Federated learning is one of the representative frameworks in distributed machine learning paradigms. However, there are several major concerns with existing federated learning paradigms. Existing distributed frameworks rely on a central server to coordinate the computing process, where such a central node may raise security concerns. Federated learning also relies on several assumptions/requirements, e.g., the independent and identically distributed (i.i.d.) data and model homogeneity. Since more and more edge devices are able to train lightweight models with local data, such models are normally heterogeneous. To tackle these challenges, in this article, we develop a blockchain-empowered federated learning framework that enables learning in a fully decentralized manner while taking the model heterogeneity and data heterogeneity into account. In particular, a federated learning framework with a heterogeneous calibration process, i.e., Model and Feature Calibration (FL-MFC), is developed to enable collaboration among heterogeneous models. We further design a two-level mining process using blockchain to enable the secure decentralized learning process. Experimental results show that our proposed system achieves effective learning performance under a fully heterogeneous environment.**

***Index Terms*—Blockchain, distributed/decentralized system, federated learning, heterogeneous features, heterogeneous models.**

I. Introduction

EDGE artificial intelligence (AI) has attracted more and more attention since edge devices are earning more and more computational resources. Many research efforts have been focusing on deploying AI over edge devices [1], [2]. Among this research, federated learning is one of the most popular approaches [3], [4], [5], [6]. Typically, federated learning needs a central server to have an initialized model. The central server will distribute this model to local workers (e.g., smart edge devices). These workers will first train the model based on their own local data sets and then send the locally trained models back to the server. The central server will aggregate collected models to obtain a well-trained global model.

However, there are several major concerns about the traditional federated learning methods. First, it requires local models to be homogeneous so that model aggregation can be easily performed over the central server. Otherwise, the central server cannot aggregate models to obtain a global model if local models are in different types or structures. However, we realize that as edge devices earn more computational power, it is more prevalent that some edge devices may already train a local model based on their own data sets. Such local models may be heterogeneous and cannot be utilized in regular federated learning. Second, to obtain a global model, traditional federated learning relies on a central party (server) to distribute, collect, and aggregate models. Such a paradigm would easily lead the system to the risk of single-point failure, which causes a critical security concern.

To address the above issues, some existing literature proposes model collaboration methods that can effectively aggregate heterogeneous models (model reuse, model rectification, etc.) [7], [8], [9], [10], [11], [12], [13]. These methods aim to aggregate the model outputs instead of model parameters. In particular, they calibrate the output of the models by slightly adjusting their parameters such that the aggregated output is accurate. However, these methods still rely on a central server to coordinate the process. To fully eliminate the center server, decentralized learning methods have been studied.

Among these methods, blockchain has been treated as an important platform to deploy the decentralized AI [14], [15], [16], [17], [18], [19], [20]. For instance, a blockchain-based multiparty learning method is proposed in [14], which enables model collaboration among heterogeneous models in a decentralized manner. However, these works only consider the issue of model heterogeneity and do not consider that heterogeneous models from different parties (devices) normally have different input shapes (i.e., feature heterogeneity). For instance, to detect specific diseases, different hospitals may have various features collected from their patients such that one hospital may record name, age, height, blood pressure, and heart rate, while another hospital may record name, age, sex, height, weight, and blood pressure. To effectively enable collaboration among models trained from individual hospitals, the algorithm should consider not only the model heterogeneity but also the feature heterogeneity. To the best of our knowledge, the investigation of model collaboration under

Manuscript received 10 May 2023; revised 19 July 2023; accepted 19 August 2023. Date of publication 4 September 2023; date of current version 6 February 2024. This work was supported by the National Science Foundation (NSF) under Grant 2245933. *(Corresponding author: Qianlong Wang.)*

The authors are with the Department of Computer and Information Sciences, Towson University, Towson, MD 21252 USA (e-mail: qwang@towson.edu; wliao@towson.edu; yguo@towson.edu; mmcguire@towson.edu; wyu@towson.edu).

Digital Object Identifier 10.1109/JIOT.2023.3311967

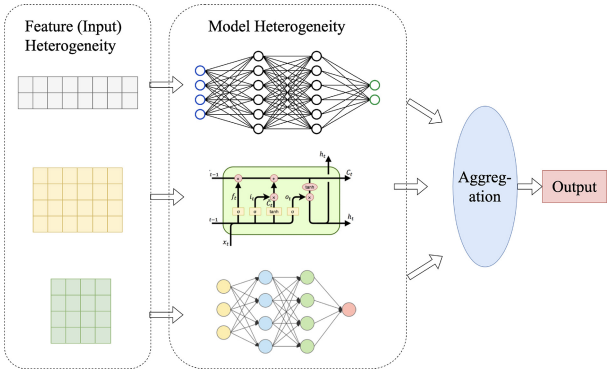

Fig. 1. Federated learning under heterogeneous environments, i.e., feature and model heterogeneity.

a fully heterogeneous environment (i.e., model and feature heterogeneity) in a decentralized manner remains an open and challenging problem.

In this article, we propose a blockchain-empowered federated learning framework, enabling model collaborations under a fully heterogeneous environment, i.e., model and feature heterogeneity. As shown in Fig. 1, our system collects different kinds of models (model heterogeneity) with different input shapes (feature heterogeneity) and aggregates them by our proposed algorithm so that effective and efficient learning performance can be achieved. In particular, we first propose a model calibration method to deal with the model heterogeneity issues. Specifically, given a data sample, we aim to aggregate the outputs of local models to obtain an accurate prediction for the sample. Our model calibration can slightly adjust the model parameters to make the system aggregated outputs accurate. By doing this, we enable collaboration among heterogeneous models. Furthermore, our model calibration method only requires a limited amount of data samples to generate effective predictions. Thus, the learning process becomes more efficient compared with traditional federated learning.

Second, to deal with feature heterogeneity, we propose a feature calibration method that enables the transfer of data samples from one model's feature space to another. Specifically, we try to learn a correlation coefficient between heterogeneous feature spaces. Correlation coefficients take advantage of the shared features between models to bridge the feature spaces. By using the optimized correlation coefficients, we are able to transfer a data sample to another feature space by estimating the value for new features. In this way, a data sample can be inputted into models with heterogeneous feature spaces. Hence, it enables model calibration among models with feature heterogeneity. Finally, we deploy both model and feature calibration processes on blockchain and design the protocols of our system.

The main contributions of this article are summarized as follows.

1) We propose a blockchain-empowered federated learning method, which enables model collaboration in a fully decentralized manner.
2) We design model and feature calibration processes to enable model collaboration under a fully heterogeneous environment, i.e., model and feature heterogeneity.
3) We deploy model and feature calibration processes on the blockchain and design the protocols in our system.
4) We evaluate the system with the real-world data set. Our results demonstrate that the proposed system is able to obtain effective performance under both model and feature heterogeneity scenarios.

The remainder of this article is organized as follows. We discuss relevant research in Section II. In Section III, we introduce the system model with assumptions. In Section IV, we present our proposed framework in detail. In Section V, we provide the performance evaluation of our proposed system. Finally, we conclude the article in Section VI.

II. Related Work

In this section, we discuss the frameworks and algorithms that are related to our study. Traditional federated learning normally relies on a central server and assumes the local models in the system to be homogeneous [3], [13], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30], [31]. In the literature, there is research focusing on removing the reliance on a central server to enable the learning process, which is known as decentralized learning, and on the other hand dealing with the heterogenous data and models. In the following, we particularly review the current blockchain-empowered decentralized learning first and then discuss the data and model heterogeneity in current federated learning.

A. Blockchain-Empowered Decentralized Learning

To deal with this issue, blockchain has been studied to enable learning in a decentralized manner. To confirm the feasibility, Rückel et al. [32] provided a proof-of-concept that blockchain technology can be combined with federated learning systems and achieve fairness, integrity, and privacy preservation. Lo et al. [33] introduced a blockchain-based method to enhance the accountability and fairness of federated learning systems. Additionally, Chen et al. [15] designed a decentralized stochastic gradient descent (SGD) algorithm over blockchain, called LearningChain, that enables a gradient aggregation process without a central server. Cui et al. [34] considered the issue of heavy communication overhead in blockchain-based federated learning and developed a communication-efficient framework by compressing communications. Shayan et al. [17] considered the security in decentralized learning and proposed Biscotti, which is a decentralized peer-to-peer (P2P) approach based on blockchain to enable multiparty machine learning. Likewise, Ma et al. [35] explored a blockchain-assisted decentralized federated learning framework, which eliminates lazy clients of the system by using designed protocols. Even though blockchain has been studied to enable the federated learning process in a decentralized manner, most of them only consider homogeneous model aggregation in federated learning. We realized that, in practice, more and more edge devices or workers in federated learning systems are able to train lightweight local models based on their local data sets. These models normally are heterogeneous. These trained local models are often ignored in

the traditional federated learning process since they cannot be utilized.

B. Data Heterogeneity in Federated Learning

There are some existing efforts to enable the federated/decentralized learning process in a heterogeneous environment. For example, Wang et al. [36] considered the convergence of federated optimization under heterogeneous/non-i.i.d data and proposed a normalized averaging method. Mendieta et al. [37] found that standard regularization methods are effective in reducing the data heterogeneity's influence and developed a method called FedAlign. Sery et al. [38] developed a convergent over-the-air federated learning (COTAF) algorithm, which precedes the local gradient at the users and scales at the server. Lin et al. [39] investigated decentralized learning limitation over data heterogeneity and proposed a momentum-based method to accelerate the training efficiency of decentralized learning. Vogels et al. [9] designed an information propagation mechanism, called RelaySum, which is to relay the original message instead of gossip averaging in the decentralized network. Horvath et al. [40] introduced a federated learning algorithm over heterogeneous clients, called Fjord. The algorithm included an ordered dropout in the model to mitigate the workload of clients with limited computational resources. Similarly, Hu et al. [41] designed an algorithm, called ADSP, that allows edge devices with more computational resources to train more epochs.

On the other hand, knowledge distillation is another promising solution for heterogeneous federated learning. For example, Zhu et al. [42] introduced a data-free knowledge distillation where the server learns generalized users' information in a data-free manner and regulates the local training process. Some other works considered secure and robust federated learning under heterogeneous data. Li et al. [43] proposed robust stochastic subgradient methods, called RSA, to enable distributed learning over heterogeneous and non-i.i.d data sets, which are shown to be resilient to Byzantine attacks. He et al. [44] proposed a gradient resampling scheme that can ensure federated learning performance under non-i.i.d data and meanwhile is resilient to Byzantine attacks. However, most of the above works focus on heterogeneous data rather than a heterogeneous model in decentralized learning. Furthermore, in many of the studies, the algorithms still rely on a central server which leads to the system not being a fully decentralized framework.

C. Model Heterogeneity in Federated Learning

Regarding model collaboration among heterogeneous models, Wu et al. [7] proposed Heterogeneous Model Reuse, HMR, to slightly adjust the pretrained heterogeneous local models in the system such that the aggregated results from models are accurate. Ye et al. [8] proposed a rectify via heterogeneous predictor mapping (REFORM) framework enabling the current model to learn from a related model with different sets of features or labels. Diao et al. [45] proposed a federated learning framework, named HeteroFL, that allows local models with different sizes in the system. Wang et al. [14] designed a decentralized multiparty learning framework under a heterogeneous model, called BEMA, that enables models to collaborate with each other by slightly adjusting their model parameters over the blockchain platform. Yu et al. [46] considered federated learning among neural networks with different structures and introduced a feature-oriented regulation method to identify and train matchable structures from them. Huang et al. [47] proposed Federated Cross-Correlation and Continual Learning, which used an unlabeled public data set to build up a generalized representation for heterogeneous models. Likewise, Li et al. [48] proposed efficient federated learning over heterogeneous mobile devices to enable flexible communication compression by balancing the energy consumption of local computing and wireless communication. However, in general, we realize there are still some drawbacks in the existing works. The heterogeneous models in some works mentioned above refer to the models with different sizes, e.g., neural networks with different number of layers, which is not a fully heterogeneous setup. Furthermore, most of the existing research did not discuss model collaboration in a fully heterogeneous environment, i.e., both model and data (feature) heterogeneity. To the best of our knowledge, studying learning frameworks in a fully decentralized and fully heterogeneous environment remains an open and challenging problem.

TABLE I
NOTATIONS IN THE SYSTEM

Indices	
N	Number of parties (users) in the system.
C	Number of classes (label for models) in the system.
d_i	Number of features of the data sample held by party i.
d_s	Number of shared features.
$d_{i'}$	Number of self-owned features of party i.
N_i	Number of data samples of party i.
Parameters	
$\mathcal{D}$	Global dataset.
$\mathcal{D}_i$	Local dataset on party i.
F_i	Local model (classifier) on party i.
f_i	Score function of F_i.
θ_i	Parameters of F_i.
F_{sys}	System aggregated classifier.
f_{sys}	System aggregated score function.
ω	The proposed secure multiparty multiclass margin.
$\mathcal{S}(y)$	Set of parties that hold data in class y.
Δ_i	Gradient of neural network model on party i.
η	Learning rate of neural network model.
X_i	Feature set of local dataset on party i.
Ψ	Correlation coefficients that are used to map a data sample from one feature space to another.

III. SYSTEM MODEL

In this section, we formally define the model and feature heterogeneity and introduce the system model. All key notations used in the article are listed in Table I.

Definition 1 (Model Heterogeneity): We consider a set of learning models that contain different model structures as the heterogeneous model set. For example, a model set could contain convolutional neural network (CNN), recurrent neural networks (RNN), and a Logistic Regression classifier. In addition, a model set containing neural networks with a different

number of layers or nodes is considered a heterogeneous model set.

Definition 2 (Feature Heterogeneity): We consider a set of data items that contain different sizes or dimensions as a heterogeneous feature set. For example, a feature set contains time-series data and images. A feature set containing images with different sizes is considered as a heterogeneous feature set.

In the system, we consider there are a total of N parties (users). Each party $i \in [1, N]$ has its own local data set $\mathcal{D}_i$ and local model (classifier) F_i. $\mathcal{D}_i = \{\mathcal{X}_i, \mathcal{Y}_i\} = \{(x_i^1, y_i^1), \ldots, (x_i^{n_i}, y_i^{n_i})\}$. x_i and y_i are the data sample and label, respectively. n_i denotes the number of data samples held by party i. It should be noted that the data sample from two parties might not have the exact same dimensions indicating feature heterogeneity in the system. In other words, the dimension of x_i may not be equal to x_j, where $i, j \in [1, N]$. Here, the elements of feature x_i and x_j may have different semantic meanings. For instance, in the hospital example, x_i includes feature of {*age, sex, height, blood pressure*}, while x_j includes feature of {*age, sex, height, weight, heart rate*}. We denote $\mathcal{D} = \{\mathcal{X}, \mathcal{Y}\}$ as the global data set, where $\mathcal{Y} = \{1, \ldots, C\}$ is the set of total C classes. A certain party i trains a local model F_i based on its local data set D_i, where $D_i \subset \mathcal{D}$.

It should be noted that the classifiers from two parties might not be the same model, which reflects the model heterogeneity of the system. For instance, the classification model can be a support vector machine (SVM), artificial neural network (ANN), CNN, or any other learning model. Since the classifier F_i is trained on only a partial set of $\mathcal{D}$, it makes it possible that the model may misclassify an unseen data sample x into a wrong class $y' \in \mathcal{Y}_i$, while its real class is $y \notin \mathcal{Y}_i$. The goal of the system is to effectively adjust the weights of the local models and to enable each party to train a more robust local model that can not only classify data from its learned space but also make predictions if the data is from an unknown space. More importantly, using the proposed method, the system will be able to utilize models $\{F_1, \ldots, F_N\}$ to make a final prediction for a specific data sample.

To generate the final system prediction for a certain data sample, we first show the output of the local model. Given a data sample x, the output of a local model f_i is

$$F_i(x) = \arg\max_{y \in \mathcal{Y}_i} f_i(\theta_i, x, y). \tag{1}$$

Here, $f_i(\theta_i, x, y)$ is the scoring function of the classifier, which is to generate a confidence score that sample x belongs to class y. θ_i is the parameter (weights) of the model, which is pretrained based on its local data set. It should be noticed that the size of unseen data x might not fit the input size of the model F_i because of the system feature heterogeneity assumption. The proposed scoring function f_i will still be able to generate an effective score given x of different sizes. The details will be elaborated in Section IV-C.

With local models, the system will generate a final prediction for the data sample x, which is listed as follows:

$$F_{\text{sys}}(x) = \arg\max_{y \in \mathcal{Y}} f_{\text{sys}}(x, y). \tag{2}$$

Here, f_{sys} is our proposed algorithm that will aggregate the scores given by all local models, i.e., $\{f_1(\theta_1, x, y), \ldots, f_N(\theta_N, x, y)\}$. The details of the score-generating algorithm will be elaborated in (5) in Section IV-B.

IV. Blockchain-Empowered Federated Learning With Model and Feature Calibration

In this section, we introduce our blockchain-empowered federated learning system in detail. Particularly, we begin with introducing the design rationale and then detail the key processes, followed by the framework implementation.

A. Design Rationale

The general system processes are as follows. We consider there are a series of pretrained local models in the system. Each of them is well-trained on its own local data set. Given a certain data sample x, we aim to utilize all the trained local models to obtain an accurate class prediction. In particular, x will be inputted into all the local models. Each model outputs a confidence score for each class. The system will aggregate all these output scores based on the proposed method, as shown in (2). We will elaborate on the aggregation function in (2) in the following sections.

Considering the model and label heterogeneity in the system, suppose y is the true class of x, it is possible that certain local models have never been trained over class y. This will directly make certain models output invalid confidence scores and might mislead the system aggregated confidence score for (x, y). To deal with this issue, we introduce the model calibration process, which aims to increase the confidence score of the local models over (x, y) and, in the meantime, decrease the confidence score of the local model over certain false classes. Hence, the accuracy of the aggregated confidence score for the system for (x, y) can be increased. The details will be elaborated in Section IV-B.

On the other hand, we consider the local models may have different input shapes (feature heterogeneity). Thus, certain data examples x cannot be directly inputted into the models. To deal with this, we introduce a feature calibration process that adopts a semantic mapping method that can effectively learn the correlations between model-shared features and model self-owned features. Based on the learned mapping correlations, one model's feature space can be projected to another one's. Hence, the data can be inputted into the model with different input dimensions. The details will be elaborated in Section IV-C. Finally, we introduce the implementation of the system over the blockchain platform and elaborate on the whole detailed system process in Section IV-D.

B. Model Calibration

To enable the model collaboration between heterogeneous models, the traditional ways (e.g., averaging the weights of models) will certainly not be effective. In the case of model heterogeneity, we aim to avoid the case that certain local models misclassify unseen data with an extremely high confidence score. Such prediction scores will largely affect the system prediction. For instance, given a data sample (x, y), where y

is the true label of the sample x, we try to avoid the case as follows:

$$f_i(\theta_i, x, y) < f_j\left(\theta_j, x, y^-\right) \quad y \in \mathcal{Y}_i, y^- \in \mathcal{Y}_j, y \notin \mathcal{Y}_j. \tag{3}$$

Here, i and j are two local models in the system. Here, model i is pretrained over label y, because $y \in \mathcal{Y}_i$. Model j is not pretrained over label y, because $y \notin \mathcal{Y}_j$. To model j, sample (x, y) is unseen data from an unknown class.

In the above situation, the model j gives a high confidence score on a wrong class y^- that belongs to its own label space $\mathcal{Y}_j$, which is even higher than the confidence score given by local model that has been trained on this class (y). Since the system is in a fully decentralized environment, there is no third authorized party that could claim model j is misclassifying a data sample. A high score on a wrong class will largely affect the system prediction over the data sample, despite what kind of score aggregation method is being used.

It should be noted that we assume all the local models are well-pretrained, meaning that the models can make the right predictions over data in their own label space. That is, given a data example (x, y), where y is the true label of x

$$f_i(\theta_i, x, y) = \max_{y \in \mathcal{Y}_i} f_i(\theta_i, x, y) \quad \forall i \in [1, N], \text{if } y \in \mathcal{Y}_i. \tag{4}$$

This is a fair assumption that all the models learn their own label space well since each of them has its own local data set. On the other hand, to generate the system prediction for a data sample x, (2) is used. In particular, we design the f_{sys} as follows:

$$f_{\text{sys}}(x, y) = \frac{1}{m} \sum_{k \xrightarrow{m} j^*} f_k(\theta_k, x, y) \tag{5}$$

$$j^* = \arg \min_{j \in \mathcal{S}(y)} \sum_{k \xrightarrow{m} j} \left| f_k(\theta_k, x, y) - f_j\left(\theta_j, x, y\right) \right| \tag{6}$$

$f_{\text{sys}}(x, y)$ represents system predicted confidence score for data x belonging to class y. $k \xrightarrow{m} j$ means the m closest value to $f_j(\cdot, \cdot, \cdot)$, where m is the number of neighbors to include for calculating the confidence score, which is a predefined parameter. $\mathcal{S}(y)$ represents the set of parties that hold data in class y. In general, (5) and (6) depict the system-predicted confidence score for (x, y) incorporates the output m of the local models instead of using only one single local model, which makes the system-predicted confidence score more resilient to bogus and abnormal values.

In the situation given by (3), it is possible that

$$f_{\text{sys}}(x, y) < f_{\text{sys}}\left(x, y^-\right) \tag{7}$$

where y and y^- are the true and wrong labels of sample x, respectively. Hence, the system misclassifies sample x into certain wrong class y^-. To solve this issue, we adopt a metric, i.e., secure multiparty multiclass (MPMC)-margin, to calibrate the models in the system [14]. Specifically, given a data sample (x, y), the secure MPMC-margin is defined as follows:

$$\omega\left(x, y, y^-\right) = \min_{y^- \in \mathcal{Y} \setminus \{y\}} f_{\text{sys}}(x, y) - f_{\text{sys}}\left(x, y^-\right) \tag{8}$$

where $\omega(x, y, y^-)$ represents the margin of confidence score obtained by the system between true class y and a false class y^-. When $\omega(x, y, y^-)$ is positive, it means that the system outputs a higher confidence score on true class y. In other words, the system makes the correct prediction for sample x. On the other hand, if it is negative, it means the system outputs a higher confidence score on a false class y^-, which indicates the system makes a wrong prediction over x. The secure MPMC margin is proven to be effective and resilient to critical attacks in a decentralized system, e.g., Byzantine attacks [14]. In this work, we similarly assume the majority of the users in the system are honest, which is a reasonable assumption.

With the secure MPMC margin, we can measure the distance between the system's predicted confidence score for the true class and certain false classes. Given a data sample (x, y), where y is the true class of x, when the secure MPMC margin is a negative value, i.e.,

$$\omega\left(x, y, y^-\right) \leq 0 \tag{9}$$

we will calibrate all $\mathcal{S}(y)$ and $\mathcal{S}(y^-)$ models. In general, the goal of model calibration, as mentioned previously, is to slightly adjust the model parameters (weights) to increase the system confidence score for x if it belongs to its true class y, i.e., $f_{\text{sys}}(x, y)$, and to decrease the system confidence score for x if it belongs to a false class y^-, i.e., $f_{\text{sys}}(x, y^-)$. Specific parameter adjustment methods can be designed for particular models. For instance, given a neural network, the gradient descent method is applied to update model parameters. In this case, we can add a positive gradient to the model parameters to increase the model's confidence score on (x, y) and, on the other hand, add a negative gradient to the model parameters to decrease the model's confidence score on (x, y^-). In our system, we assume that all the local models support an online updating process. That means each model can be slightly updated to increase or decrease the confidence score on a certain data sample.

C. Feature Calibration

So far, by using model calibration, we can adjust the model output in order to increase the accuracy of system-generated predictions. Our model calibration also works under conditions where model heterogeneity exists. In the following, we consider the input side of the system. The decentralized system may contain models with different input sizes, i.e., feature heterogeneity, which is common in reality [8]. For such models, the model calibration cannot be applied as the data sample cannot be used as input for both models. For instance, as previously mentioned, two hospitals may collect different features from their patients and train models to detect certain diseases, though most of these features are common. Even if only a small portion of features are self-owned by a certain model (model-specific), the model calibration cannot be applied. Hence, these models cannot collaborate to increase the system performance, which is obviously a waste of resources.

To effectively utilize the model with feature heterogeneity, we propose a feature calibration method, which aims to calibrate the space of the input features. We first assume that

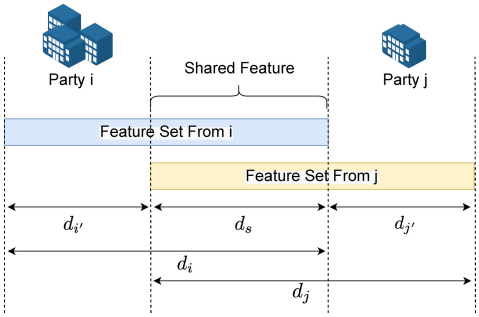

Fig. 2. Features collected from two certain parties i and j.

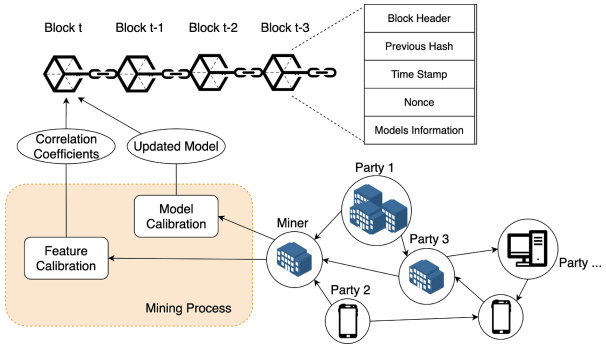

Fig. 3. Architecture of blockchain-empowered federated learning with the model and feature calibration.

there are shared features existing between models. Hence, considering two models, each of their features can be decomposed into model-shared and model self-owned features. We believe such an assumption is fair and common in practice. For instance, as we mentioned, two hospitals may try to train models to predict the presence of a certain disease using slightly differing sets of features. Although most of the features used in prediction might be the same, some of them might be only collected from one hospital but not others. Enlightened by semantic mapping introduced in work [8], we aim to learn the correlations between the heterogeneous feature space. The general idea of the feature calibration is to learn the correlations between model-shared and model self-owned features. With the learned correlations, we can estimate the model's self-owned features based on the model-shared features. Hence, we are able to rebuild a feature space that contains model-shared features and model self-owned features from both models.

Considering two local models, e.g., model i and j, with the feature X_i and X_j. Suppose X_i and X_j have the dimension (the number of features) of d_i and d_j, respectively. We denote the dimension of the shared feature between X_i and X_j as d_s. We also denote dimension of model self-owned features of X_i and X_j as $d_{i'}$ and $d_{j'}$, respectively. As shown in Fig. 2, X_i and X_j can be denoted as $[X_i^{d_{i'}} \in \mathbb{R}^{N_i \times d_{i'}}, X_i^{d_s} \in \mathbb{R}^{N_i \times d_s}]$ and $[X_j^{d_s} \in \mathbb{R}^{N_j \times d_s}, X_j^{d_{j'}} \in \mathbb{R}^{N_j \times d_{j'}}]$. Here, N_i and N_j are the number of the data samples of party i and j, respectively. $X_i^{d_s}$ and $X_j^{d_s}$ are the model-shared features with the same semantic meaning. Next, we aim to learn correlations between shared features, i.e., $X_i^{d_s}$ and $X_j^{d_s}$, and model self-owned features, i.e., $X_i^{d_{i'}}$ and $X_j^{d_{j'}}$. With such learned correlations, model-shared features can be used to estimate feature value for model self-owned features and hence transfer to another feature space.

To learn the correlations between shared and self-owned features, we define a correlation coefficient, i.e., Ψ, for each model. Given two models, i and j as mentioned above, we learn correlation coefficients Ψ_i and Ψ_j, respectively

$$\arg\min_{\Psi_i} ||X_i^{d_{i'}} - X_i^{d_s}\Psi_i||_F^2 + \lambda \sum_{m=1}^{d_{i'}} ||\Psi_{i,m}||_0 \tag{10}$$

$$\arg\min_{\Psi_j} ||X_j^{d_{j'}} - X_j^{d_s}\Psi_j||_F^2 + \lambda \sum_{m=1}^{d_{j'}} ||\Psi_{j,m}||_0. \tag{11}$$

Here, $\Psi_i \in \mathbb{R}^{d_s \times d_{i'}}$ and $\Psi_j \in \mathbb{R}^{d_s \times d_{j'}}$ are learned coefficients that map the shared features to the self-owned features. The second term of (10) and (11) are regularization terms, which are controlled by a predefined value λ. We aim to have the coefficients as sparse as possible so that the learned correlations focus more on the generalized patterns [49], [50]. We use orthogonal matching pursuit (OMP) to solve (10) and (11) efficiently.

With the learned correlations, the feature of model i, X_i, is transferred to the model j's space, which is denoted as $X_{j\leftarrow i}$

$$X_{j\leftarrow i} = \left[X_i^{d_s} \in \mathbb{R}^{N_i \times d_s}, X_i^{d_s}\Psi_j \in \mathbb{R}^{N_i \times d_{j'}}\right]. \tag{12}$$

Similarly, the feature of model j, X_j, is transferred to the model i's space, which is denoted as $X_{i\leftarrow j}$

$$X_{i\leftarrow j} = \left[X_j^{d_s}\Psi_i \in \mathbb{R}^{N_j \times d_{i'}}, X_j^{d_s} \in \mathbb{R}^{N_j \times d_s}\right]. \tag{13}$$

Hence, the data sample from one model can be transferred to another model's space as long as they have shared features. In this way, the model calibration can be applied to models under feature heterogeneity.

D. Framework Implementation

We now elaborate on how our system is implemented on the blockchain, which enables a multiparty learning process in a fully decentralized manner. The main framework implementation process is demonstrated in Fig. 3. As shown in the figure, each user (party) in the system could be a smart edge device, personal computer, or an institution holding certain workstations, where each of them holds a local model. In particular, our system consists of users of two roles, i.e., regular users and miners. The regular users are considered as parties who hold pretrained local models and local data sets. The regular users who join the system aim to increase the performance of their local models and further use local models in the system to make a more accurate prediction for unseen data. Specifically, the regular user in the system will broadcast his local model, which will be registered/recorded on the blockchain by a miner. Meanwhile, every regular user will find out valid calibration data samples from his local data set and broadcast them. On the other hand, the miner will use such samples to perform the mining process, i.e., model and feature calibration, to update the models on the blockchain.

The updated model parameters will be written in a new block created by the miner. By doing this, the aggregation of local models' predictions can be effective. Next, we elaborate on the protocols in our system for two roles.

Regular users are assumed to hold local models and local data sets and aim to utilize the system for effective predictions. Regular users are allowed to perform the following operations.

1) *Broadcast Local Models:* The regular users are encouraged to broadcast their local models in the system. Hence, the miner can use them for the model calibration process.
2) *Broadcast Portion of Local Data Set:* The regular users are encouraged to broadcast a portion of their local data set in the system. Based on the broadcasted data set, the miner can implement the feature calibration process and select certain valid data samples for the model calibration process. Such a process may cause concern about data privacy leakage. However, it should be noted that only a limited amount of data samples are needed in the system for the feature and model calibration process, which is confirmed via simulation results. Therefore, the compromise of data privacy is considered to be moderate.
3) *Classify Unseen Data:* The regular users are entitled to use the local models in the system for classifying unseen data samples, which is (2).

Miners are expected to implement the feature and model calibration processes so that regular users can effectively use the system to classify unseen data. Any regular user is encouraged to implement the miner's role. Certain rewards/incentives will be given to the miner. In particular, the activities the miners are expected to do are as follows.

1) *Feature Calibration:* Given the broadcasted local models and the corresponding data sets, for any two models in the system that contain shared features, the miner calculates (10) and (11) to obtain optimized correlation coefficients. Once a new coefficient is obtained, it will be recorded in a new block of the blockchain and the corresponding miner will be rewarded. Since the feature calibration method is public, every obtained correlation coefficient can be verified by any user in the system. Hence, it ensures the authenticity of feature calibration.
2) *Model Calibration:* Given the correlation coefficients obtained from the feature calibration process, a data sample (x, y) can be inputted into any local model so long as the model input space has shared features with x. In model calibration, the miner needs first to find a valid data sample for model calibration and then use that data sample to implement the process. The details are as follows.
   a) *Find Valid Data Sample:* The miner picks certain data sample (x, y) from the broadcasted data sets and tries to find out a class y^- such that (9) is satisfied. If found, a data sample (x, y, y^-) will be broadcasted in the system, which can be used for model calibration. Similar to the feature calibration process, once such a sample is found by the miner, it will be recorded in the block, and certain rewards will be given to the corresponding miner. Since (x, y, y^-) is from the public data set, the validity of the sample can also be verified by any user in the system.
   b) *Online Update Process:* Given an obtained sample (x, y, y^-), the miner performs a positive and negative online update process for $\mathcal{S}(y)$ and $\mathcal{S}(y^-)$ models, respectively. The updated model parameters will also be recorded in the new block. Similar to other processes, the calibrated models can also be verified by any user in the system, and rewards will be given to the miner.

Algorithm 1 Blockchain-Empowered Federated Learning With the Model and Feature Calibration

1: **Regular User:**
2: Each regular user i broadcasts its model information, including a subset of its local dataset and model parameters.
3: Implement Eq. (2) to classify unseen data.
4: **Miner:**
5: Feature Calibration:
6: Implement Eqs. (10) and (11) to obtain optimized correlation coefficients.
7: Model Calibration:
8: Find data sample (x, y, y^-) satisfy constraints Eq. (9).
9: Implement positive and negative online updates for $\mathcal{S}(y)$ and $\mathcal{S}(y^-)$ models to increase and decrease model confidence score on (x, y) and x, y^-, respectively.

Since every aforementioned process can be verified by any user, they are considered as the proof of work (PoW) for the miner. We also demonstrate the framework implementation process in Algorithm 1. In the algorithm, we summarize the main processes for both regular users and miners. In particular, the regular user is encouraged to broadcast its local model and a subset of its local data set. In the meantime, the regular user can utilize the models on the blockchain to classify the unseen data. The miner then performs the feature and model calibration processes for model updating. So far, we have introduced the implementation of the blockchain. In conclusion, our system now enables collaborations among heterogeneous models with heterogeneous input space.

V. Performance Evaluation

In this section, we investigate the learning performance of the proposed federated learning with model and feature calibration. In particular, we separately conduct simulations considering the system contains heterogeneous models and heterogeneous features. In the case of heterogeneous models, we include different deep learning models in the system and test the proposed model calibration method on MNIST data set [51]. In the case of heterogeneous features, we use synthetic data and apply both feature and model calibration methods to verify the system's effectiveness. In both settings, comparisons with results from existing methods have been provided.

A. Classification with Heterogeneous Models

1) Data Preprocessing: To simulate the multiparty setting, we separate the data set into different parties with different

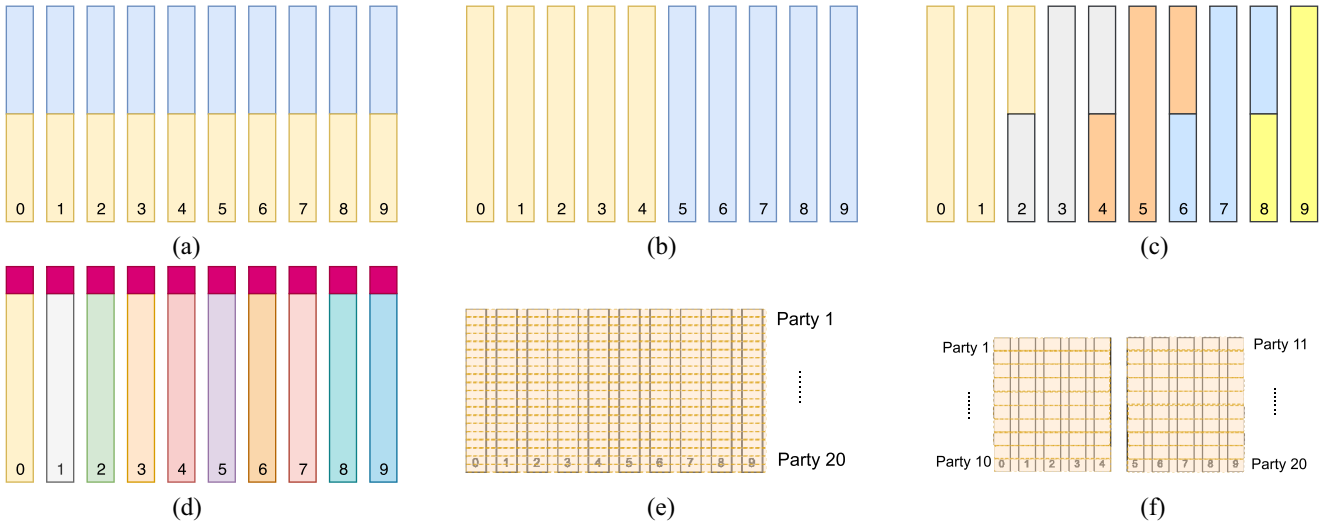

Fig. 4. Data distributions in different simulation cases. Each color represents the local data on a party. There is no data overlap between different parties. In (d), the red color denotes the shared public data that every party can access. In (e) and (f), the data are separated into 20 parties. In the former one, all the ten classes' data are evenly separated into 20 parties. In the latter one, the data distribution is skewed, where each party only has data from a total of five classes. (a) Two parties, case I. (b) Two parties, case II. (c) Five parties. (d) Ten parties. (e) 20 parties, case I. (f) 20 parties, case II.

data distributions, according to Fig. 4. We extend a similar setting in [14], where six cases are formulated. These cases vary from 2 to 20 parties. Some of the cases are data unified distributed; some are skewed distributed, in which some parties may never see the data from some specific classes.

2) Implementation (Benchmark): We compare our method with a number of existing methods. First, we evaluate our method by using a single model on the whole data set. In other words, a single model is trained on the whole MNIST data set. We adopt LeNet-5 [51], which is a popular CNN model. Then, we adopt a popular distributed multiparty learning method, HMR, as a comparison model [7]. In HMR, a central trusted server collects the local data and models from all parties. The server checks models over the union of all the data to find the calibration samples by using an MPMC margin and generates the final system output by using a max-model predictor.

Model Deployment and Calibration: In test cases of 2-party and 5-party, all the parties are deployed by the LeNet-5 model. In test cases of 10-party and 20-party, heterogeneous models, including both ANNs and CNNs, are deployed on different parties. In particular, we adopt different popular CNN models in the system, which include LeNet-5, ResNet, and VGG [51], [52], [53]. Each of these models is deployed on a certain party and trained on its own local data set. However, since each party only has a partial data set, these models may not perform well on the class that is missing in its local data set. Next, the system implements a model calibration process to calibrate the models' output. Specifically, the miner in the system will select a valid calibration sample (x, y, y^-), which makes the $\omega(x, y, y^-)$ in (8) a negative value. Once such a sample is verified, the miner performs positive and negative online update processes for $\mathcal{S}(y)$ and $\mathcal{S}(y^-)$, respectively. In this way, the system prediction score $f_{\text{sys}}(x, y)$ and $f_{\text{sys}}(x, y^-)$ can be increased and decreased, respectively. Finally, once a data sample (x, y) is given, the system generates the prediction score based on (2).

3) Simulation Results and Discussion: The comparisons of accuracy over designed data distribution cases are shown in Fig. 5. It can be observed from the figure that the proposed federated learning with model and feature calibration method (FL-MFC) reaches a similar performance as HMR in all cases, which demonstrates the effectiveness of the proposed framework. It should be noted that the convergence speed of FL-MFC is slower than HMR. This is because the proposed framework is designed for the untrusted decentralized environment. The secure aggregation algorithm in the proposed framework ensures that the model calibration method is resilient to bogus and malicious models, which is proved in [14]. This aggregation ensures the performance; in the meantime, slightly degrades the performance convergence speed. We believe this is fair and acceptable in decentralized systems, especially when there is no authority party exists.

Additionally, when it comes to 20 parties in cases e and f, since the data amounts on each party are decreased compared with the previous cases, it takes a longer time to calibrate the models in the system for both HMR and FL-MFC. Another finding is that case f finally reaches a higher accuracy than case e, even though they are both 20 parties. We believe this is caused by data distribution. In case e, every party has data samples from all classes from 0 to 9. In case f, each party only has data samples from partial classes. For instance, party 1 only has data from classes 0 to 4, while party 11 has data from classes 5 to 9. However, in this case, each party has fewer classes to learn. In other words, party 1 in case f needs to train the model over classes from 0 to 4, while party 1 in case e needs to train all 10 classes. Furthermore, party 1 in case e has fewer data samples for each class than party 1 in case f. This causes that party 1 in case e may not be able to be well-trained over the local data set. On the other hand, party 1 in case f would outperform party 1 in case e over classes from 0 to 4. Based on this, case f outperforms case e after the model calibration. In general, to generate high accuracy, we should

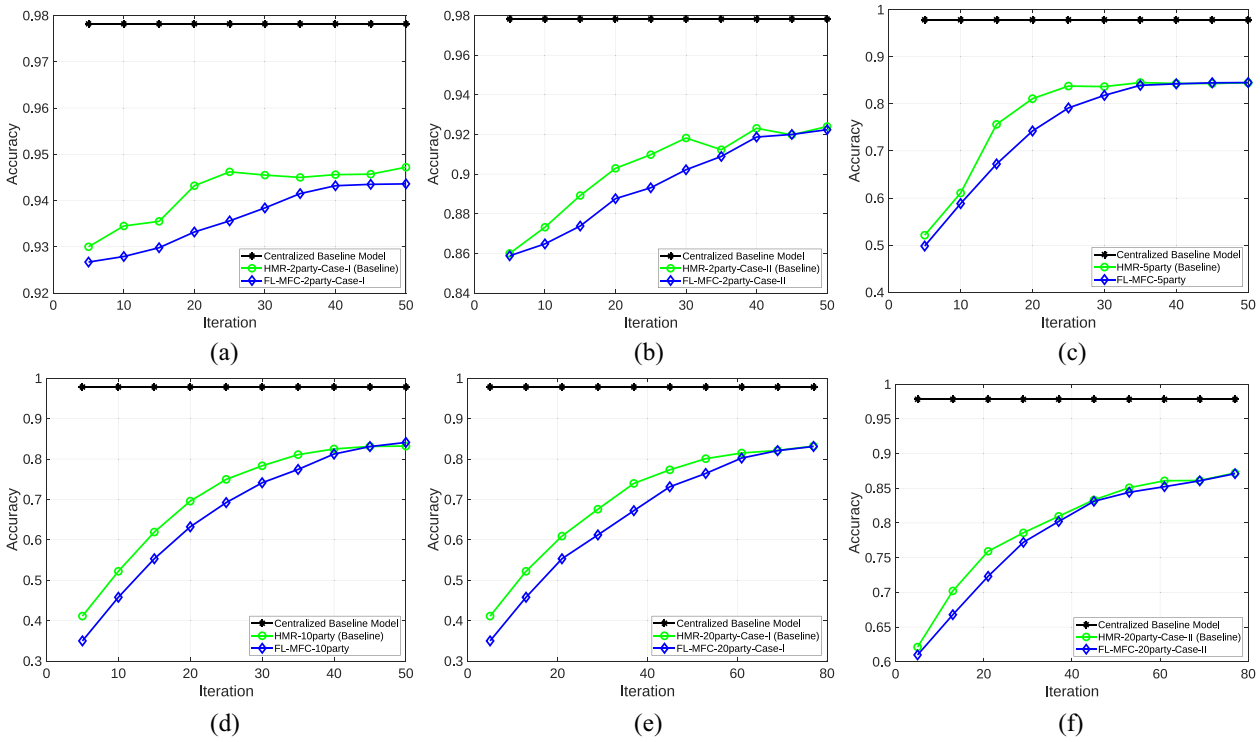

Fig. 5. Comparison results of accuracy on different data distribution cases. (a) Two parties, case I. (b) Two parties, case II. (c) Five parties. (d) Ten parties. (e) 20 parties, case I. (f) 20 parties, case II.

be inclined to have each party trained on each class it has as well as possible rather than let each party have data samples from more classes.

B. Classification With Heterogeneous Features

In this section, we investigate the performance of the proposed framework considering there are heterogeneous features that exist in different parties of the system. In particular, we consider a scenario in which there are two parties, e.g., parties 1 and 2, in the system. These two parties have local data sets with heterogeneous features and models. We implement the feature and model calibration process in this section to investigate the effectiveness of the proposed system.

1) Data Preprocessing: We adopt five data sets from the UCI machine learning repository in this section [54]. For each data set, we partition the data samples, split the attributes into two parties, and investigate the performance of feature + model calibration. Specifically, we consider there are two parties in the system. Given a data set, we partition the data set into half and half. The first 50 percent of data samples are assigned to party 1, and the rest is assigned to party 2. Additionally, all attributes of the data set are randomly split into three parts.

The portion of the data for party 1 self-owned attributes, party 2 self-owned attributes, and shared attributes are 35%, 35%, and 30%, respectively. In this way, there are only 30% of overlapping features between two parties. In addition, we use 70 % of local data are used for training and 30 % for testing. In summary, party 1 contains the first half of the data set and 65% of attributes, while party 2 has the second half and 65% of attributes. 30% of attributes are shared by both parties.

2) Implementation: In this simulation, we deploy an SVM model [55] on both parties 1 and 2. Each model is trained using 70% of its local data set, as mentioned above. Since the attributes owned by each party are public information. Parties 1 and 2 can learn correlation coefficients Ψ_1 and Ψ_2, respectively, based on the shared and self-owned features through (10) and (11). Once done, each party will broadcast their model parameters and correlation coefficients, i.e., Ψ_1 and Ψ_2 in the system.

Hence, party 1 can reconstruct a data sample, $X_{2\leftarrow 1}$, by using (12), that fits the input space of the model on party 2. Similarly, party 2 can reconstruct its local data set to party 1's input space. So far, the feature calibration process has been completed. Both parties can input their local data samples to the other's model and generate predictions without exposing their local data set to the other party. Then, a similar process in the previous simulations can be conducted to perform model calibration. Specifically, each party finds a data sample (x, y, y^-) that makes $\omega(x, y, y^-)$ in (8) a negative value and broadcasts it in the system. Once the miner in the system verifies that, they will perform positive and negative online updates to increase and decrease the corresponding model's prediction score on the sample, respectively. It should be noted that the results demonstrate that less than 1% of the amount of the global data set is enough to make the performance converge in the model calibration process. This indicates the number of shared data samples is very limited, which leads

TABLE II
COMPARISON RESULTS OF ACCURACY ON DATA SETS. THE LAST ROW REPRESENTS THE TIMES OF WIN OR LOSS ON THE PROPOSED FRAMEWORK, I.E., FL-MFC VERSUS OTHERS

	FL-MFC	HFA	OTL	BEMA
Colic	**0.722**	0.652	0.707	0.686
German Credit Data	0.676	0.680	**0.720**	0.675
Spambase	**0.818**	0.722	0.803	0.770
SPECTF	0.776	0.801	**0.934**	0.329
Waveform	**0.660**	0.626	0.639	0.368
FL-MFC (W / L)	N/A	3 / 2	3 / 2	5 / 0

to moderate privacy leakage. Finally, each party obtains the system prediction score for an unseen data sample through (2).

For the comparison models, we adopt two heterogeneous transfer learning methods, called HFA [56] and OTL [57]. The HFA method transfers the local training spaces of each model to an augmented space. In particular, it needs to incorporate data from the other party in the current party training process, which causes data exposure to a certain extent. OTL trains models in an online manner. It builds a co-regularized regularizer to make predictions from two feature spaces. In addition to HFA and OTL, we test a naive setup in which there is still an SVM on each party, but no feature and model calibration processes are implemented. In other words, considering party 1 and party 2, their SVM models are only trained on their local data sets. Then, given an unseen data sample, we directly apply the score aggregation function, (2), to obtain the system prediction score.

3) Results and Discussion: The comparison results are shown in Table II. The highest accuracy is in bold. Through five real-world data sets, the proposed framework, FL-MFC, outperforms the existing methods, HFA and OTL. Especially compared with the BEMA algorithm, the FL-MFC can effectively calibrate the heterogeneous feature spaces among different parties and further enhance the system's accuracy. Without feature calibration, the BEMA only conducts the model collaboration among homogeneous input space. If most of the models in the system have different input spaces, the BEMA algorithm's performance is largely degraded, as shown in SPECTF and Waveform in Table II. Overall, results from both Sections V-A and V-B indicate that the FL-MFC can, first, reach similar performance with existing methods when there are only heterogeneous models and, second, outperform existing ones when there are both heterogeneous models and features.

VI. CONCLUSION

In this article, we have proposed a blockchain-empowered federated learning system that enables model collaboration in a fully heterogeneous environment, including both heterogeneous models and feature spaces. In particular, we have designed model and feature calibration processes to enable heterogeneous models with heterogeneous input spaces to collaborate with each other to effectively improve their performance. With the proposed aggregation algorithm, the models in the system are able to output high-accuracy results on unseen data samples. Performance evaluation over machine learning and real-world data sets has been conducted to validate the efficacy of model and feature calibration, respectively. The comparison results have demonstrated the effectiveness of the proposed system under a heterogeneous environment.

ACKNOWLEDGMENT

Any opinions, findings, and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect the views of the National Science Foundation.

REFERENCES

[1] W. G. Hatcher and W. Yu, "A survey of deep learning: Platforms, applications and emerging research trends," *IEEE Access*, vol. 6, pp. 24411–24432, 2018.
[2] F. Liang, W. G. Hatcher, W. Liao, W. Gao, and W. Yu, "Machine learning for security and the Internet of Things: The good, the bad, and the ugly," *IEEE Access*, vol. 7, pp. 158126–158147, 2019.
[3] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020.
[4] P. Tian, W. Liao, W. Yu, and E. Blasch, "WSCC: A weight-similarity-based client clustering approach for non-IID federated learning," *IEEE Internet Things J.*, vol. 9, no. 20, pp. 20243–20256, Oct. 2022.
[5] Z. Chen, W. Liao, P. Tian, Q. Wang, and W. Yu, "A fairness-aware peer-to-peer Decentralized learning framework with heterogeneous devices," *Future Internet*, vol. 14, no. 5, p. 138, 2022. [Online]. Available: https://www.mdpi.com/1999-5903/14/5/138
[6] Z. Chen, P. Tian, W. Liao, and W. Yu, "Zero knowledge clustering based adversarial mitigation in heterogeneous federated learning," *IEEE Trans. Netw. Sci. Eng.*, vol. 8, no. 2, pp. 1070–1083, Apr.–Jun. 2021.
[7] X.-Z. Wu, S. Liu, and Z.-H. Zhou, "Heterogeneous model reuse via optimizing multiparty multiclass margin," in *Proc. Int. Conf. Mach. Learn.*, 2019, pp. 6840–6849.
[8] H.-J. Ye, D.-C. Zhan, Y. Jiang, and Z.-H. Zhou, "Heterogeneous few-shot model rectification with semantic mapping," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 43, no. 11, pp. 3878–3891, Nov. 2021.
[9] T. Vogels et al., "RelaySum for decentralized deep learning on heterogeneous data," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 28004–28015.
[10] D. Gao, Y. Liu, A. Huang, C. Ju, H. Yu, and Q. Yang, "Privacy-preserving heterogeneous federated transfer learning," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2019, pp. 2552–2559.
[11] C. Xu, Y. Qu, Y. Xiang, and L. Gao, "Asynchronous federated learning on heterogeneous devices: A survey," 2021, *arXiv:2109.04269*.
[12] K. Pillutla, Y. Laguel, J. Malick, and Z. Harchaoui, "Federated learning with heterogeneous data: A superquantile optimization approach," 2021, *arXiv:2112.09429*.
[13] T. Nishio and R. Yonetani, "Client selection for federated learning with heterogeneous resources in mobile edge," in *Proc. IEEE Int. Conf. Commun. (ICC)*, 2019, pp. 1–7.
[14] Q. Wang, Y. Guo, X. Wang, T. Ji, L. Yu, and P. Li, "AI at the edge: Blockchain-empowered secure multiparty learning with heterogeneous models," *IEEE Internet Things J.*, vol. 7, no. 10, pp. 9600–9610, Oct. 2020.
[15] X. Chen, J. Ji, C. Luo, W. Liao, and P. Li, "When machine learning meets blockchain: A decentralized, privacy-preserving and secure design," in *Proc. IEEE Int. Conf. Big Data (Big Data)*, 2018, pp. 1178–1187.
[16] Y. Tian, T. Li, J. Xiong, M. Z. A. Bhuiyan, J. Ma, and C. Peng, "A blockchain-based machine learning framework for edge services in IIoT," *IEEE Trans. Ind. Informat.*, vol. 18, no. 3, pp. 1918–1929, Mar. 2022.
[17] M. Shayan, C. Fung, C. J. M. Yoon, and I. Beschastnikh, "Biscotti: A blockchain system for private and secure federated learning," *IEEE Trans. Parallel Distrib. Syst.*, vol. 32, no. 7, pp. 1513–1525, Jul. 2021.
[18] Y. Lu, X. Huang, Y. Dai, S. Maharjan, and Y. Zhang, "Blockchain and federated learning for privacy-preserved data sharing in industrial IoT," *IEEE Trans. Ind. Informat.*, vol. 16, no. 6, pp. 4177–4186, Jun. 2020.

[19] J. Weng, J. Weng, J. Zhang, M. Li, Y. Zhang, and W. Luo, "DeepChain: Auditable and privacy-preserving deep learning with blockchain-based incentive," *IEEE Trans. Dependable Secure Comput.*, vol. 18, no. 5, pp. 2438–2455, Sep./Oct. 2021.

[20] L. Yin, J. Feng, S. Lin, Z. Cao, and Z. Sun, "A blockchain-based collaborative training method for multi-party data sharing," *Comput. Commun.*, vol. 173, pp. 70–78, May 2021.

[21] P. Kairouz et al., "Advances and open problems in federated learning," *Found. Trends® Mach. Learn.*, vol. 14, nos. 1–2, pp. 1–210, 2021.

[22] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, p. 12, 2019.

[23] K. Bonawitz et al., "Towards federated learning at scale: System design," 2019, *arXiv:1902.01046*.

[24] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," 2018, *arXiv:1806.00582*.

[25] X. Liu, H. Zhao, M. Pan, H. Yue, X. Li, and Y. Fang, "Traffic-aware multiple mix zone placement for protecting location privacy," in *Proc. IEEE INFOCOM*, 2012, pp. 972–980.

[26] V. Smith, C.-K. Chiang, M. Sanjabi, and A. S. Talwalkar, "Federated multi-task learning," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 4424–4434.

[27] A. Lalitha, S. Shekhar, T. Javidi, and F. Koushanfar, "Fully decentralized federated learning," in *Proc. 3rd Workshop Bayesian Deep Learn. (NeurIPS)*, 2018, pp. 1–9.

[28] Y. Esfandiari et al., "Cross-gradient aggregation for decentralized learning from non-iid data," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 3036–3046.

[29] K. Hsieh, A. Phanishayee, O. Mutlu, and P. Gibbons, "The non-iid data quagmire of decentralized machine learning," in *Proc. Int. Conf. Mach. Learn.*, 2020, pp. 4387–4398.

[30] S. Li, T. Zhou, X. Tian, and D. Tao, "Learning to collaborate in decentralized learning of personalized models," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 9766–9775.

[31] X. Liang, A. M. Javid, M. Skoglund, and S. Chatterjee, "Asynchrounous decentralized learning of a neural network," in *Proc. IEEE Int. Conf. Acoust., Speech Signal Process. (ICASSP)*, 2020, pp. 3947–3951.

[32] T. Rückel, J. Sedlmeir, and P. Hofmann, "Fairness, integrity, and privacy in a scalable blockchain-based federated learning system," *Comput. Netw.*, vol. 202, Jan. 2022, Art. no. 108621.

[33] S. K. Lo et al., "Toward trustworthy AI: Blockchain-based architecture design for accountability and fairness of federated learning systems," *IEEE Internet Things J.*, vol. 10, no. 4, pp. 3276–3284, Feb. 2023.

[34] L. Cui, X. Su, and Y. Zhou, "A fast blockchain-based federated learning framework with compressed communications," *IEEE J. Sel. Areas Commun.*, vol. 40, no. 12, pp. 3358–3372, Dec. 2022.

[35] C. Ma et al., "When federated learning meets blockchain: A new distributed learning paradigm," *IEEE Comput. Intell. Mag.*, vol. 17, no. 3, pp. 26–33, Aug. 2022.

[36] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "A novel framework for the analysis and design of heterogeneous federated learning," *IEEE Trans. Signal Process.*, vol. 69, pp. 5234–5249, 2021.

[37] M. Mendieta, T. Yang, P. Wang, M. Lee, Z. Ding, and C. Chen, "Local learning matters: Rethinking data heterogeneity in federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 8397–8406.

[38] T. Sery, N. Shlezinger, K. Cohen, and Y. C. Eldar, "Over-the-air federated learning from heterogeneous data," *IEEE Trans. Signal Process.*, vol. 69, pp. 3796–3811, 2021.

[39] T. Lin, S. P. Karimireddy, S. U. Stich, and M. Jaggi, "Quasi-global momentum: Accelerating decentralized deep learning on heterogeneous data," 2021, *arXiv:2102.04761*.

[40] S. Horvath, S. Laskaridis, M. Almeida, I. Leontiadis, S. Venieris, and N. Lane, "FjORD: Fair and accurate federated learning under heterogeneous targets with ordered dropout," in *Proc. Adv. Neural Inf. Process. Syst.*, vol. 34, 2021, pp. 12876–12889.

[41] H. Hu, D. Wang, and C. Wu, "Distributed machine learning through heterogeneous edge systems," in *Proc. AAAI Conf. Artif. Intell.*, vol. 34, 2020, pp. 7179–7186.

[42] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *Proc. Int. Conf. Mach. Learn.*, 2021, pp. 12878–12889.

[43] L. Li, W. Xu, T. Chen, G. B. Giannakis, and Q. Ling, "RSA: Byzantine-robust stochastic aggregation methods for distributed learning from heterogeneous datasets," in *Proc. AAAI Conf. Artif. Intell.*, vol. 33, 2019, pp. 1544–1551.

[44] L. He, S. P. Karimireddy, and M. Jaggi, "Byzantine-robust learning on heterogeneous datasets via resampling," 2020, *arXiv:2006.09365*.

[45] E. Diao, J. Ding, and V. Tarokh, "HeteroFL: Computation and communication efficient federated learning for heterogeneous clients," 2020, *arXiv:2010.01264*.

[46] F. Yu et al., "Heterogeneous federated learning," 2020, *arXiv:2008.06767*.

[47] W. Huang, M. Ye, and B. Du, "Learn from others and be yourself in heterogeneous federated learning," in *Proc. IEEE/CVF Conf. Comput. Vis. Pattern Recognit.*, 2022, pp. 10143–10153.

[48] L. Li, D. Shi, R. Hou, H. Li, M. Pan, and Z. Han, "To talk or to work: Flexible communication compression for energy efficient federated learning over heterogeneous mobile edge devices," in *Proc. IEEE Conf. Comput. Commun.*, 2021, pp. 1–10.

[49] H. Wang, F. Nie, and H. Huang, "Multi-view clustering and feature learning via structured sparsity," in *Proc. Int. Conf. Mach. Learn.*, 2013, pp. 352–360.

[50] H. Wang, F. Nie, W. Cai, and H. Huang, "Semi-supervised robust dictionary learning via efficient l-norms minimization," in *Proc. IEEE Int. Conf. Comput. Vis.*, 2013, pp. 1145–1152.

[51] 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.

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

[53] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2014, *arXiv:1409.1556*.

[54] D. Dua and C. Graff. "UCI machine learning repository." 2017. [Online]. Available: http://archive.ics.uci.edu/ml

[55] R.-E. Fan, K.-W. Chang, C.-J. Hsieh, X.-R. Wang, and C.-J. Lin, "LIBLINEAR: A library for large linear classification," *J. Mach. Learn. Res.*, vol. 9, pp. 1871–1874, Jun. 2008.

[56] W. Li, L. Duan, D. Xu, and I. W. Tsang, "Learning with augmented features for supervised and semi-supervised heterogeneous domain adaptation," *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 36, no. 6, pp. 1134–1148, Jun. 2014.

[57] P. Zhao, S. C. Hoi, J. Wang, and B. Li, "Online transfer learning," *Artif. Intell.*, vol. 216, pp. 76–102, Nov. 2014.