2025 IEEE 45th International Conference on Distributed Computing Systems (ICDCS) | 979-8-3315-1723-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICDCS63083.2025.00046

FedShapleX: Shapley Value Driven Context-Aware Model-Heterogeneous Federated Learning

Jifeng Chen, Haibo Zhang, Amanda Barnard
ANU College of Engineering, Computing & Cybernetics, Australian National University, Canberra, Australia
{u7903697, haibo.zhang, amanda.s.barnard}@anu.edu.au

***Abstract*—Model Heterogeneous Federated Learning(MHFL) builds on traditional Federated Learning (FL) to better leverage the knowledge and data distributed across hardware-heterogeneous devices. Among various heterogeneous FL approaches, the Partially Training (PT)-based methods are one of the most promising approaches, which extract submodels from the global model for local training. However, existing state-of-the-art(SOTA) methods lack effective guidance for updating the global model, making it challenging to handle the Non-IID data distribution and maintain generalization across clients. To guide the update of the global model to mitigate the impact of Non-IID data and enhance the generalization of the global model, we proposed FedShapleX: Shapley Value Driven Context-Aware Submodel Extraction for Model-Heterogeneous Federated Learning. In this work, we first proposed a Parameter-based Class-Specific Shapley Value (PCSV), which quantifies each client's class-specific contribution to the global model, providing a measure of how effectively the local knowledge is utilized. Leveraging the contribution assessment, we further develop a Reinforcement Learning-aided Large Neighbourhood Search Algorithm (RL-LNS) algorithm, which optimizes the submodel extraction scheme based on context-aware contribution information, thereby guiding the global model update more effectively. Leveraging the actor-critic scheme, the RL-LNS combines the strengths of Large Neighbourhood Search (LNS) and Reinforcement Learning (RL), improving the LNS's search efficiency while simplifying the design of RL policies. To validate the RL-LNS, we have compared the FedShaplex against the state-of-the-art (SOTA) partial training-based approach MHFL, the global model performance, and its average accuracy on clients' datasets.**

***Index Terms*—Federated Learning, Contribution Evaluation, Shapley Value, Incentive Mechanism**

I. Introduction

Federated Learning (FL) is a typical distributed machine learning technique that enables multiple clients to collaboratively train a machine learning model under the coordination of a central server. Instead of sharing raw data, clients exchange model parameters, such as gradients or weights, achieving joint model optimization while preserving data privacy. [1] Therefore, FL has been widely applied in scenarios where data privacy is critical, or data transmission bandwidth is limited, such as financial fraud detection and collaborative sensing in satellite constellations. [2]

In FL, heterogeneity is an unavoidable and challenging issue affecting the training process categorized into three types: **1) Computational heterogeneity** Computational heterogeneity refers to the differences in computational capabilities across participants caused by hardware, resource allocation, and computing environment variations. **2) Communicational heterogeneity** refers to differences in communication capabilities among participants, such as varying network bandwidths, latency, or reliability. **3) Data heterogeneity** refers to the differences in data distributions across participating clients. FL involves decentralized data that is typically non-IID and unbalanced. These heterogeneities are fundamental challenges in FL, stemming from the decentralized, real-world characteristics of clients, and they can significantly impact the convergence and performance of the global model.

These heterogeneity challenges motivated the field of MHFL. Recent SOTA research has proposed various methods to address the issues caused by these heterogeneities, which can be broadly categorized into three main approaches: Knowledge Distillation(KD)-based approaches [3], [4], Mutual Learning-based approaches [5], [6], and Partial Training(PT)-based approaches [7], [8].

In hardware heterogeneous scenarios, particularly in edge computing, PT-based approaches are regarded as the most promising solution due to their adaptability to computational resource constraints. However, regardless of the approach used, they have yet to address data heterogeneity effectively. This issue is primarily reflected in parametric model divergence and other factors impacting convergence and global model performance [9]. In the PT-based model heterogeneous scenario, the challenges become more pronounced. Firstly, due to the differences in data classes held across the client, a sub-model highly tailored to specific classes might be assigned to a client with largely irrelevant data, introducing a deterioration into the Global Model. Secondly, the sub-models assigned to individual clients tend to over-fit their local dataset, which introduces parametric model divergence and ultimately hinders the convergence of the global model.

To enhance the adaptability of the PT-based approaches, we should consciously guide the updating direction of the global model. The main challenges are: **1) Evaluating client contributions**: Accurately evaluating the contribution of each individual client without relying on public datasets or incurring high computational or communicational costs. **2) Guided Federated Learning Scheme**: Proposing a guided MHFL scheme to prevent the global model from drifting due to data heterogeneity.

To address the issues above, we introduce a Parameter-based Class-Specific Shapley Value (PCSV) to quantify local clients' contribution, thereby measuring the efficiency of local knowledge utilization. By leveraging the class-specific con-

2575-8411/25/$31.00 ©2025 IEEE
DOI 10.1109/ICDCS63083.2025.00046

tribution as context information, we propose the Reinforcement Learning-aided Large Neighbourhood Search (RL-LNS) method to optimize submodel extraction. This approach enhances knowledge utilization, mitigates global model drift, and maximizes the effective use of local computational resources.

Overall, the main contributions of this paper can be summarized as follows.

- *Parameter-based Class-Specific Shapley Value (PCSV):* We proposed a parameter-based method to evaluate the class-specific contribution of each client, serving as a more computationally efficient and privacy-protected alternative to the traditional Shapley Value definition.
- *Reinforcement Learning-aided Large Neighbourhood Search Algorithm (RL-LNS):* We developed a novel RL-aided LNS algorithm to determine the optimal sub-model extraction strategy for each client dynamically.
- *FedShapleX:* We proposed a FedShapleX, a Shapley Value-driven context-aware submodel extraction model-heterogenous federated learning, using PCSV and RL-LNS to guide the training of the global model consciously.

II. Related Works

Individual Contribution Evaluation in Federated Learning: Computing the Shapley Value in FL is challenging due to the computation of the utility function v and the traversal through all the client subsets. From this point of view, state-of-the-art (SOTA) research can be categorized into utility function approximation and model approximation approaches.

In recent years, numerous utility function approximation methods have been proposed to replace the evaluation of training a new model aggregated from the relevant client subset. Wang *et al.* proposed a contribution evaluation algorithm named FedSV, where the server randomly samples a group of clients to approximate the utility function v [10]. However, since the FedSV assigned zero contributions to client sets not involved in the current session, the clients with the same data distribution are not guaranteed to be evaluated for the same contribution. To address this, Fan *et al.* proposed an improved version of the FedSV named ComFedSV, using a low-rank matrix completion model to approximate the utility function v as a utility matrix based on parameter and loss information [11]. Based on ComFedSV, Fan *et al.* also proposed VerFedSV, specially tailored for vertical FL [12]. While these approaches effectively estimate the utility function, they lack generalizability across different FL systems and still incur high computational complexity while evaluating individual contributions.

Numerous model approximation approaches have been proposed to simplify the traversal of all client subsets, often leveraging the gradient information. Liu *et al.* introduce the GTG-Shapley, which uses the marginal gain information to enhance computational efficiency using within-round and between-round truncation [13]. Meanwhile, a widely accepted and used work proposed by Xu *et al.* named Cosine Gradient Shapley Value (CGSV), which quantifies individual contributions based on the similarity between local gradients and the globally aggregated gradient [14]. Tastan *et al.* presented a class-specific individual contribution evaluation method named as Class-specific Shapley Value (CSSV), using the gradient of the classifier to provide a more granular evaluation of class-specific contributions [15]. These researches provide an efficient way of estimating the Shapley value without traversal through all the client subsets. However, as they are fully dependent on the gradient information, it's potential that the optimizer with specific search strategies will influence the evaluated contribution results.

Model-Heterogeneous Federated Learning: In recent years, MHFL has come in three general approaches: partial training, mutual learning, and knowledge distillation methods. Using the model-heterogeneous approaches, these methods aim to fully utilize the computational resources and the knowledge distributed to different local clients.

In mutual learning approaches, the servers distribute and aggregate homogeneous models to the clients, and the local clients use a heterogeneous model for mutual learning or fine-tuning. Shen *et al.* proposed a mutual learning scheme, where the local clients train a heterogeneous model for better local performance and use the homogeneous global model for the knowledge sharing [16]. Yi *et al.* integrated the Low-Rank Adaptation (LoRA) into the mutual learning scheme. However, as these methods require two models to be trained on the local client, the additional computational cost is not friendly to edge devices [5].

Knowledge distillation is also a promising strategy introduced into Federated Learning. Li *et al.* propose FedMD, a heterogeneous Federated Learning via Model Distillation [3]. As knowledge distillation usually depends on the public dataset, Luo *et al.* proposed DFRD, a date-free robustness distillation for heterogeneous Federated Learning [4]. However, the knowledge distillation methods will introduce rather the usage of the public dataset or additional computation cost, which is running contrary or making the federated learning challenging for the edge devices.

Compared with the approaches mentioned above, the partial training strategies are more resource-efficient in terms of computational requirements. Wen *et al.* proposed Federated Dropout, as the name indicates, introduced the dropout strategy to randomly extract the sub-model from the global model [17]. Alam *et al.* proposed FedRolex, which uses a rolling scheme to extract the sub-model from the global model [8]. Horvath *et al.* proposed the FjORD, a static scheme tailoring the sub-model fitting to the client's capacity [7]. Diao *et al.* proposed HeteroFL, which extracts submodels with different capacities and aggregates a global inference model [18]. While the partial training approaches significantly emphasize computational capacity, they face notable challenges in non-IID scenarios. Without a deliberate mechanism to guide the update direction of the global model, it would be unavoidable that the Non-IID data distribution will influence the convergence and the global model performance significantly.

III. Problem Formulation

In the SOTA Partial-Training(PT)-based MHFL, such as HeteroFL [18] and FjORD [7], the overall process can be typically divided into two key stages: **Sever Model Distribution** and **Client Model Training**. The centralized server will first extract and distribute the sub-models. Then, the clients will train the received sub-model on the local data distribution for the approaching global aggregation.

Thus, the PT-based Model Heterogeneous FL Problem addressed in our proposed work can be modeled as two intertwined subproblems:

1) **Model-Heterogeneous FL Subproblem**: The collaborative optimization of the global model parameters by the server and clients.
2) **Sub-model Extraction Subproblem**: The server leverages context and environment information to extract and distribute appropriate sub-models to clients, aiming to improve convergence and overall performance. As illustrated in Fig. 1, the sub-models are extracted from the global model on the server and distributed to the respective clients.

A. Model-Heterogeneous FL Subproblem

In the Model-Heterogeneous FL scenario, a group of clients are separately training models locally aimed at aggregating a global model that minimizes the loss function among all these clients. In a FL system, we consider a total of K communication rounds and N clients, where the set of clients is denoted as the set $\mathcal{N} = \{n_1, n_2, \ldots, n_N\}$. Each local client i can observe the local dataset defined as $D_i = \{d_{i,1}, d_{i,2}, \ldots, d_{i,k}\}$ from the environment, where $d_{i,k}$ representing the data for client i observed in the communication round k. Collectively, the local datasets across all clients are denoted as $\mathcal{D} = \{D_1, D_2, \ldots, D_N\}$. The parameters of the global model will be denoted as Θ. With these definitions, the global objective of the Partial Training (PT)-based MHFL Optimization Problem is definite as Eq. 1.

$$\begin{aligned} \min_{\Theta} F(\Theta) &\triangleq \sum F_i'(\Theta), \quad \forall i \in \mathcal{N} \\ F_i'(\Theta) &\triangleq \sum \ell(\theta_{i,k}, d_{i,k}), \theta_{i,k} \in \Theta, d_{i,k} \in D_i \end{aligned} \tag{1}$$

$F(\Theta)$ and $F_i'(\Theta)$ represent the global optimization objective and the local optimization objective, respectively. In Eq.1, the goal is to minimize the sum of the local objective functions across all local clients. The sub-model parameters extracted for client i at the round k decided by the server from the **Sub-model Extraction Problem** are denoted as $\theta_{i,k} \in \Theta$. Then, the local optimization objective can be formulated as Eq. 2.

$$\begin{aligned} \min_{\Theta} F_i'(\Theta) &\triangleq \sum \ell(\theta_{i,k}, d_{i,k}) \\ &\forall k \in [0, K], \theta_{i,k} \in \Theta, d_{i,k} \in D_i \end{aligned} \tag{2}$$

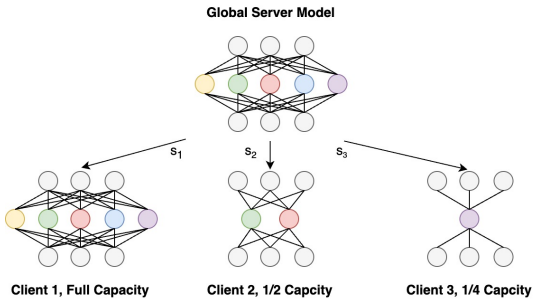

Fig. 1. Submodel Extraction Example

B. Sub-model Extraction Subproblem

The Sub-model Extraction Problem aims to optimize the model extraction scheme at each training round to improve the FL system's convergence and performance. As the proposed work aims to fully utilize the computational resources under the hardware heterogeneous scenario, the sub-models are extracted based on the computational capacity of the clients. We define the computational capacity of the clients as $\mathcal{A} = \{\alpha_1, \alpha_2, \ldots, \alpha_N\}$, which also represents the proportion of neurons extracted from each hidden layer in global model Θ for each client.

In our proposed work, a continuous extraction scheme is proposed inspired by FedRolex [8], incorporating a shift mechanism driven by optimizing the Sub-model Extraction Subproblem. The sub-model extraction subproblem provides a set of solutions for the communication round k, represented as $S_k = \{s_1, s_2, \ldots, s_N\}$, which denotes the shift for the sub-model extraction. For the sub-model $\theta_{i,k}$ assigned to client i in round k, each layer of the sub-model is defined as $\theta_{i,k}^j$, where $j \in \mathcal{J} = \{1, 2, \ldots, J-1\}$ represents the index of the hidden layers in a J layer neural network. Similarly, within the global model, each layer of the global model is denoted as Θ^j with $|\Theta^j|$ neurons. In the sub-model extraction scheme S_k, each shift s_i is in the range of $[0, \max(|\Theta_j|)], \forall j \in \mathcal{J}$. When the shift value is no larger than the maximum number of neurons in a deep neural network, it could produce different sub-model extraction schemes. During the model extraction stage, sub-models are extracted for each layer by applying the shift s_i with the proportion of α_i, which corresponds to the computational capacity of the client i. Based on these definitions, the extraction of layer j for the sub-model assigned to client i in round k can be formulated as shown in Eq. 3.

$$\theta_{i,k}^j = \begin{cases} \{s_i, s_i+1, \ldots, \\ \quad s_i + \lfloor \alpha_i |\Theta^j| \rfloor - 1\}, & \text{if } s_i + \lfloor \alpha_i |\Theta^j| \rfloor \leq |\Theta^j|, \\ \{s_i, s_i+1, \ldots, |\Theta^j| - 1\} \cup \\ \quad \{0, 1, \ldots, s_i + \lfloor \alpha_i |\Theta^j| \rfloor - 1 - |\Theta^j|\}, & \text{else.} \end{cases} \tag{3}$$

In Fig.1, we present an example involving three clients with computation capacities $\mathcal{A} = \{1, \frac{1}{2}, \frac{1}{4}\}$. Given a sub-model extraction scheme $S_k = \{s_1 = 1, s_2 = 2, s_3 = 5\}$, the sub-

model are extracted following Eq.3. The resulting sub-models are illustrated in Fig.1 illustrated.

Since the Non-IID data distribution causes the local clients to train models in divergent directions, it is essential to introduce a mechanism to guide the training and aggregation procedure of the global model. Given that the accuracy of the entire dataset reflects the model's performance, we can use the global model accuracy to guide the sub-model extraction scheme to steer the global model's improvement in a targeted direction. Based on this concept, the optimization problem could be formulated as Eq.4.

$$\begin{aligned} \arg\min_{S_k} \quad & \mathcal{L}(\mathcal{D}, \Theta) \\ \text{s.t.} \quad & \theta_{i,k}^j = \text{Extract}(\Theta^j, s_i), \\ & \theta_{i,k} = \sum_j \theta_{i,k}^j, \\ & \Theta = \text{Aggregate}(\theta_{1,k}, \theta_{2,k}, \ldots, \theta_{N,k}), \\ & k \in [0, K], \quad i \in \mathcal{N}, \quad s_i \in S_k \end{aligned} \tag{4}$$

In Eq. 4, $\mathcal{L}(\mathcal{D}, \Theta)$ represents the loss of the global model tested on the entire dataset. The optimization problem aims to find the optimal solution S_k for sub-model extraction for each communication round k to minimize global loss. However, this is an idealized assumption. In typical FL scenarios [19], the data is decentralized, and knowledge is shared exclusively through model parameter aggregation. Using the global model's performance on the entire dataset to guide local training contradicts the intuition behind FL, where local data privacy is a central principle.

In this context, mechanisms are required to evaluate the global contribution of each client independently of the entire dataset, reflecting the utilization of the knowledge distributed to each client. After each aggregation, the contribution of each client in the communication round k is denoted as $\Phi_k = \{\phi_1, \phi_2, \ldots, \phi_N\}$. This set indicates whether a client is contributing to or hindering the global model's improvement direction in the previous communication round based on their distributed sub-models. By maximizing each client's individual contributions, we can ensure that each client is aligned with the global model's improvement direction while still fitting the local data. However, maximizing the contributions in an individual communication round only guarantees that the clients are learning in a specific direction. Still, it doesn't guarantee that this direction aligns with the global model's optimal loss value. Nevertheless, by iteratively optimizing each client's contribution for each communication round, the sub-model extraction scheme can be refined to best utilize the knowledge from individual clients, ultimately guiding the improvement direction to the ground truth as closely as possible. With this concept, the sub-model extraction subproblem can be reformulated as Eq.5.

$$\begin{aligned} \arg\max_{S_k} \quad & \{\phi_1, \phi_2, \ldots, \phi_N\} \\ \text{s.t.} \quad & \theta_{i,k}^j = \text{Extract}(\Theta^j, s_i), \\ & \theta_{i,k} = \sum_j \theta_{i,k}^j, \\ & \Theta = \text{Aggregate}(\theta_{1,k}, \theta_{2,k}, \ldots, \theta_{N,k}), \\ & \phi_i = \text{Evaluate}(\theta_{i,k}, \Theta), \\ & k \in [0, K], \quad i \in \mathcal{N}, \quad s_i \in S_k \end{aligned} \tag{5}$$

In Eq.5, the Sub-model Extraction Problem is now fully dependent on the evaluation between the local client model and the global aggregate model. By decoupling from the entire dataset, this approach maintains the integrity of the original MHFL framework, eliminating the need for centralized access to the dataset.

IV. Proposed Method

To solve the abovementioned problem, we proposed a novel FL scheme called FedShapleX, which uses the Parameter-based Class-Specific Shapley Value (PCSV) to evaluate individual contribution. Then, a Reinforcement Learning-aided Large Neighbourhood Search (RL-LNS) algorithm will utilize the evaluation result as the environment information to decide on the sub-model extraction scheme in the next round.

A. Parameter-based Class-Specific Shapley Value (PCSV)

In section III, we have discussed that in the Non-IID scenario, each client fits the sub-model to local data, causing the global model to drift in different directions for improvement. This dynamic creates a competitive interaction among clients, resembling a game-theoretic scenario in which the Shapley value offers a principled approach to distributing the value of a coalition to individual participants. The trivial definition for calculating the Shapley Value calculation in FL is defined in Eq. 6. To compute the contribution of client i, the algorithm iterates through all the subsets of the client set, excluding client i, and calculates the average accuracy of global models aggregated from these subsets.

$$\phi_i = \sum_{C \subseteq \mathcal{N} \setminus \{i\}} \frac{|C|!(n - |C| - 1)!}{n!} \left(F(C \cup \{i\}) - F(C) \right) \tag{6}$$

However, the trivial definition of the Shapley Value is challenging to apply directly in FL. Firstly, the trivial definition is highly dependent on the public dataset, which needs the public dataset to test the accuracy of each global model aggregated from these client subsets. Secondly, the trivial definition demands considerable computational resources for evaluation, as the contribution of each client must be assessed across all possible subsets of clients.

The SOTA approaches addressing these challenges are generally gradient-based and not directly applicable to our scenario. Firstly, these methods focus solely on gradients rather than the parameters. Due to the influence of the optimizer, the

actual update direction for each client might deviate from the direction opposite to the gradient. This discrepancy means that these approaches may fail to reflect the contribution accurately except for gradient-based FL, such as FedSGD [19]. Secondly, for MHFL, the dimensionality of the gradient vectors might differ across clients, introducing additional unfairness into the contribution evaluation.

With these challenges, we proposed the Parameter-based Class-specific Shapley Value (PCSV). Unlike gradient-based methods, PCSV focuses on model parameters, which offer a more stable and interpretable basis for contribution evaluation. Since gradients propagate backward from the last layer to the first, the magnitude of parameter changes tends to diminish in earlier layers. Thus, to better capture the direction of model improvement, we focus on the linear classifier, where parameter changes are more noticeable and directly impact classification performance.

The classifier in the J-layer Deep Neural Network can be viewed as a function mapping the input $|\Theta^{J-1}|$-dimensional feature embedding into a $|\mathcal{M}| = |\Theta^{J}|$-dimensional logits space, where $\mathcal{M} = 1, 2, 3, \ldots, M$ represents the indices of the classification categories. The parameters of the global model's classifier at round k are denoted as $W_k \in \mathcal{R}^{|\Theta^{J-1}|\times|\mathcal{M}|}$, while the corresponding parameters in the local client i at round k are represented as $w_{i,k} \in \mathcal{R}^{|\theta^{J-1}_{i,k}|\times|\mathcal{M}|}$. Since the classifier's output directly corresponds to the classification categories, we can treat its parameters as $|\mathcal{M}|$ separate columns, each representing the weights associated with a specific classification. Accordingly, the global parameters related to classification m are represented as $W'_{k,m} \in W_k$, and the corresponding local parameters are represented as $w'_{i,k,m} \in w_{i,k}$.

In each communication round, the server first collects the parameter updates from clients, which are represented as $\hat{w}'_{i,k,m} \in \hat{w}_{i,k}$. After the aggregation, the global update direction for category m can be calculated as $V_{\Theta,m} = W_k - W_{k-1}$, and the local update direction can be calculated as $v_{\theta_i,m} = \hat{w}'_{i,k,m} - w'_{i,k,m}$. If the local update direction $v_{\theta_i,m}$ is aligned with the global update direction $V_{\Theta,m}$, it indicates that the client is contributing positively to the global model. Conversely, if the local direction deviates noticeably from the global direction, it shows that the local client is pushing the global model towards local data distribution, potentially causing model drift.

We denote the Class-specific individual contribution matrix for communication round k be denoted as $\Gamma_k = \{\varphi_{i,m} | \forall i \in \mathcal{N}, \forall m \in \mathcal{M}\}$. Then, the overall and class-specific contribution of each client i can be computed according to Eq.7. Here, Π is a project operator, which projects the global updating direction onto a subspace where only the parameters associated with the neuron selected by the local client are retained. As shown in Eq.7, the class-specific contribution is determined by calculating the cosine similarity between the local and global updating directions. Then, the overall individual contribution ϕ_i is obtained by averaging the class-specific contributions. The overall PCSV algorithm is shown in the Alg.1.

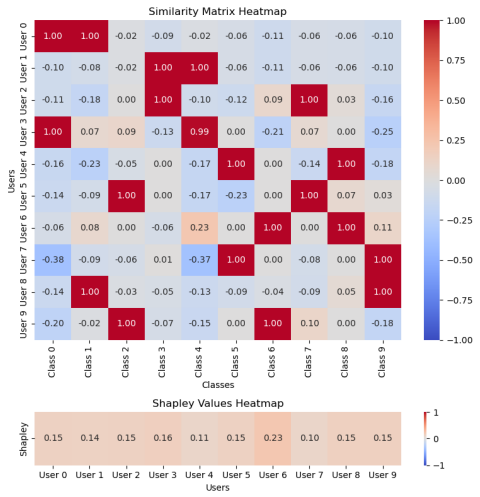

Fig. 2. Experimental Demonstration contrast the true Shapley Value approach and the PCSV evaluation on Cifar10 dataset with 10 clients in a Model-Heterogeneous Federated Learning (MHFL) system.

$$\varphi_{i,m} = \cos(v_{\theta_i,m}, \Pi_{\theta^J_{i,k}}(V_{\Theta,m})),$$
$$\phi_i = \frac{1}{M}\sum_{m=1}^{M} \varphi_{i,m} \tag{7}$$

PCSV Example Considering a MHFL scenario, where there are $N = 10$ clients trained on CIFAR-10 dataset ($M = 10$). To simulate a Non-IID data distribution, the dataset is partitioned such that each client only has access to samples from two randomly assigned classes out of the total ten classes. We apply the PCSV algorithm to compute the Class-specific Shapley Value and utilize both the standard Shapley Value definition and the PCSV Class-specific Shaply Value matrix to assess the overall individual contribution of each client.

As shown in Fig.2, a dark red cell with a large positive value indicates a significant positive contribution, while a dark blue color with a smaller negative value represents a noticeable negative contribution. The results reveal that the clients contribute significantly to the classification tasks related to the data they possess. Specifically, each client contributes substantially to the two categories of data assigned to them, reflecting the characteristics of the Non-IID data distribution. This also indicates that the PCSV approximation closely aligns with the actual contribution.

B. Reinforcement Learning-aided Large Neighbourhood Search Algorithm (RL-LNS)

Between each communication round of FL, the PCSV algorithm evaluates the Class-specific Shapley Value and the overall Shapley Value, representing the environmental states of the MHFL problem. Based on this environmental information, the Sub-model Extraction Subproblem determines the sub-model extraction strategy for each client in the communication

Algorithm 1 Parameter-based Class-Specific Shapley Value (PCSV)

Require: Global Parameters W_k, Local Parameters $w_{i,k}$, Updated Local Parameters $w'_{i,k}$

Ensure: Individual Contributions Set Φ_k, Class-Specific Contribution Matrix Γ_k

$V_{\Theta,m} = W_k - W_{k-1}$
for $m = 1$ to M **do**
  $v_{\theta_i,m} = \hat{w}'_{i,k,m} - w'_{i,k,m}$
end for
for $i = 1$ to N **do**
  for $m = 1$ to M **do**
    $\varphi_{i,m} = \cos(v_{\theta_i,m}, \Pi_{\theta^J_{i,k}}(V_{\Theta,m})), \varphi_{i,m} \in \Gamma_k$
  end for

end for
return Φ_k, Γ_k

round. Firstly, the sub-model extraction sub-problem can be formulated as a Markov Decision Process (MDP).

MDP Modeling: The Markov Decision Process is described as a 4-tuple $(\textit{State}, \textit{Action}, \textit{Probability}, \textit{Reward})$. Following this tuple, the Sub-model Extraction Subproblem can be formulated as follows:

- *State*: The state of the MHFL system consists of the sub-model extraction information and the Class-Specific Shapley Value, represented as a 2-tuple $state_k = (S_k, \Phi_k)$. As previously discussed, the feasible space of the S_k could have a size of $\max(|\Theta_j|)]^N, \forall j \in \mathcal{J}$, based on the range of the shift value. Meanwhile, each ϕ_i in Φ_k can take any continuous value within the range of $[-1, 1]$. Combining these factors together creates an infinite-size state space.
- *Action*: The Sub-model Extraction leading to the next state can take any feasible S_{k+1}. This action depends solely on the previous state without any additional constraints. As discussed earlier, the action space could also have a size of $\max(|\Theta_j|)]^N, \forall j \in \mathcal{J}$.
- *Probability*: The probability $P(state'|state, S_{k+1})$ of transitioning the current state to the next state is determined through the interactive sampling within the MHFL system. Heuristic algorithms can be employed to approximate this function, leveraging experience replay to improve the fitting process.
- *Reward*: Since the overall performance of the global model is not directly observable, the reward is defined as the overall individual Shapley value calculated by the PCSV algorithm, denoting Φ_k. This reward directly reflects the extent to which the knowledge of each individual client is utilized to contribute to the global model. Each overall individual contribution in ϕ_k lies within the range of $[-1, 1]$.

Since this problem is modeled as a Markov Decision Process, it exhibits the property of sequential decision-making. While Reinforcement Learning methods have the potential to solve such problems, the unique characteristics of this problem pose significant challenges to the direct application of RL methods.

Firstly, it is highly challenging to model the transition probability $P(state'|state, S_k + 1)$. Since the next state is determined by the feedback from the FL system, it is difficult to represent the state transition probability function using mathematical models. This complexity makes model-based Reinforcement Learning methods difficult to apply in such scenarios. Moreover, the state space is infinite, which poses a significant challenge for traditional RL methods like Q-learning, as they rely on tabular representation to store all the possible states and experiences. Given these characteristics, it is intuitive that model-free reinforcement learning like Double Deep Q-Network (DDQN) [20] should be considered in such scenarios.

However, applying DDQN also comes with its own set of challenges, particularly related to the complexity of the deep Q-network and its convergence. Traditional Deep Q-Networks (DQNs) use the size of the action space size as the dimension of the output layer. In our scenario, a complex action embedding may be required to represent the entire action space. Additionally, the cost of obtaining new experiences is high, and the Q network lacks generalization across different MHFL settings. Consequently, training a complex Deep Q-network with limited experience becomes a significant challenge.

To address the abovementioned problems, we proposed the Reinforcement Learning-aided Large Neighbourhood Search (RL-LNS), which employs an actor-critic scheme to improve the solution process. In this scheme, the Large Neighbourhood Search (LNS) plays the actor responsible for generating and updating candidate solutions. The DDQN component acts as the critic, evaluating the quality or fitness of these solutions. By combining the solution evolution capability of LNS with the evaluation capability of the RL critic, this approach aims to achieve more efficient convergence and faster solution evolution in MHFL settings.

RL-LNS Framework: Fig.3 illustrates the overall framework of the RL-LNS algorithm framework. On the left-hand side, the Large Neighbourhood Search (LNS) actor is illustrated. The actor begins by generating a heuristic solution using a heuristic solution generator. On the right-hand side of the figure is the DDQN critic, giving feedback to the candidate solution obtained from the RL-LNS actor for solution evolution. This iterative feedback process between the actor and critic continues to refine the sub-model extraction strategies for each client.

LNS Actor: The Large Neighbourhood Search (LNS) framework combines greediness with exploration. In each iteration, LNS preserves a relatively good portion of the solution while destroying the remaining part for reconstruction. As illustrated in Fig.3, if no current solution exists, the actor utilizes the *Heuristic Solution Generator* to produce an initial feasible solution S_k. This solution is then encoded using the *Positional Solution Embedder*, which incorporates both the

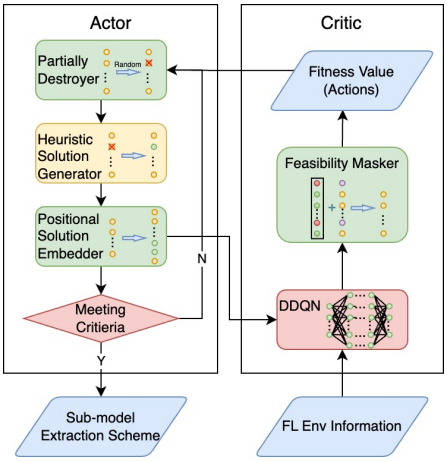

Fig. 3. The Reinforcement Learning-aided Large Neighbourhood Search algorithm framework

solution and the indices of the participating clients.

Guided by feedback from the DDQN Critic, which utilizes the class-specific individual contribution Γ_k as the environment information, the feedback is represented as the fitness value $\mathcal{F}(S_k, \Gamma_k) = \{f(s_1, \Gamma_k), f(s_2, \Gamma_k), \ldots, f(s_N, \Gamma_k)\}$, the *Partially Destroyer* eliminates a fixed proportion of the solution, focusing on components with low fitness value. The preserved portion is subsequently treated as a constraint set, represented as $\mathcal{C}$, for the *Heuristic Solution Generator* during solution reconstruction. After the reconstruction, since the sub-model extraction aims to optimize the contribution of all the clients, resulting in a multi-objective optimization problem, the reconstructed candidate solution, represented as $S'_k = \{s'_1, s'_2, \ldots, s'_N\}$, will be updated as the new solution S_k if and only if the reconstructed candidate solution Pareto dominates the original solution. After a fixed number of iterations, the candidate solution is returned as the optimized sub-model extraction scheme for the current communication round. The corresponding algorithm is described in Alg.2.

Algorithm 2 Reinforcement Learning-aided Large Neighbourhood Search Algorithm (RL-LNS)

Require: Class-specific individual contribution Γ_k
Ensure: Sub-model Extraction Scheme S_k

```
while Criteria not met do
    if S_k exist then
        C = PartiallyDestroy(S_k, F(S_k, Γ_k))
        S'_k = Heuristic(C)
        if F(S_k, Γ_k) ⪰ F(S'_k, Γ_k) then
            S_k = S'_k
        end if
    else
        S_k = Heuristic()
    end if
end while
return S_k
```

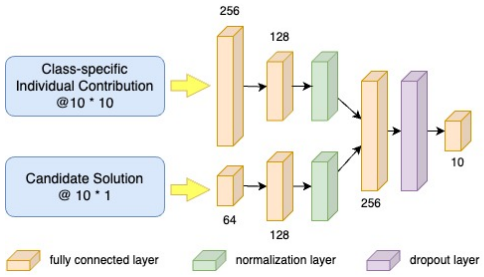

Fig. 4. Deep Q Network Design

DDQN Critic: The DDQN will use the Class-specific contribution Γ_k as the environment information, combined with the solution generated from the *LNS actor*, the DDQN will output the fitness vector, represented as $\mathcal{F}(S_k, \Gamma_k)$. We designed the Deep Q-Network illustrated in Fig.4 to achieve this.

The Deep Q Network (DQN) proposed in our approach is a multi-layer perceptron (MLP) with two input branches and a merged output branch. Given the significant difference in the range of candidate solutions compared to the context information, which is the class-specific individual contributions specifically, we employ two separate input branches to encode the features of the candidate solution and the environmental context. Once extracted, the features are normalized separately and concatenated to be fed into the merged branch. A dropout layer is introduced before the final linear layer to mitigate overfitting and ensure stable output with faster convergence.

To enable the Deep Q-Network (DQN) to act as a critic, it evaluates the current candidate solution in conjunction with the environmental information. Specifically, the Deep Q-Network predicts the value of $\mathcal{F}(S_k, \Gamma_k)$, which reflects the fitness of the current candidate solution in the context of each client's class-specific contribution. To achieve this, Temporal-Different(TD) Learning is integrated into the training procedure of the DDQN, using the PCSV estimated Shapley value as the reward. Additionally, to ensure stable updates of Q-network and to prevent overestimation of fitness value $\mathcal{F}(S_k, \Gamma_k)$, we adopted the DDQN architecture, which consists of a target Q network Q_{target} for prediction and an evaluation Q network Q_{eval} for network updates. Then, the TD error is introduced into the training of the Deep Q-network for the backpropagation and updating the weights update of Q_{eval}, defined as Eq.8.

$$\mathcal{L}_{TD} = \Phi_k + \gamma * \mathcal{F}_{Q_{target}}(S_{k+1}, \Gamma_{k+1}) - \mathcal{F}_{Q_{eval}}(S_k, \Gamma_k) \quad (8)$$

During each communication round (except for the first), the FL system stores the current sub-model extraction scheme S_k, the current Class-specific individual contribution Γ_k, the previous sub-model extraction scheme S_{k-1}, the previous Class-specific individual contribution Γ_{k-1}, and the overall individual contribution Φ_k into the experience buffer as a tuple $(S_{k-1}, \Gamma_{k-1}, \Phi_k, S_k, \Gamma_k)$.

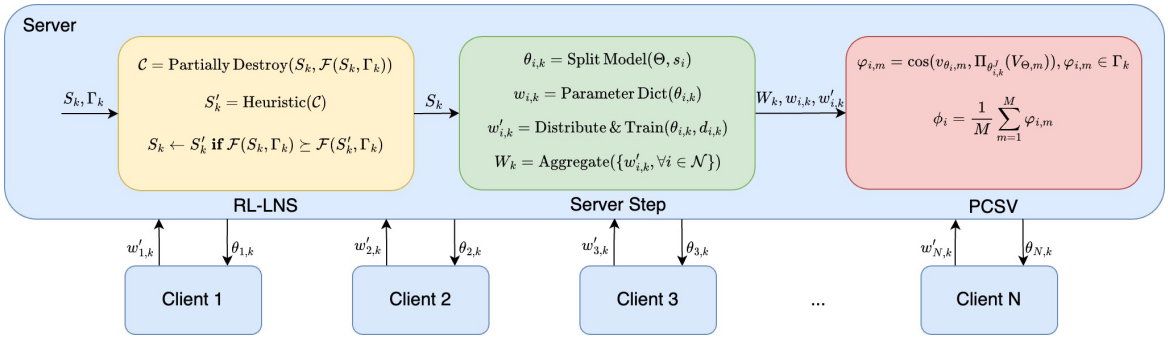

Fig. 5. Overview of the FedShapleX Algorithm

During the experience replay, the sampled experience is firstly used to calculate the TD loss $\mathcal{L}_{TD}$. Once the Deep Q Network Q_{eval} is updated, the Deep Q Network Q_{target} employs a soft target update strategy to adjust its parameters in alignment with the updated Deep Q Network Q_{eval}, following $Q_{target} = \tau * Q_{eval} + (1-\tau) * Q_{target}$. This approach ensures smoother updates and improved training stability.

C. FedShapleX: Shapley Value Driven Context-Aware Sub-model Extraction

Based on the proposed PCSV and RL-LNS algorithms, we propose FedShapleX, a Shapley Value-Driven, Context-Aware Submodel Extraction framework for MHFL. By leveraging the contribution information derived from PCSV, FedShapleX can guide the training by utilizing each client's knowledge effectively. This approach enhances both convergence speed and model accuracy.

Feature Alignment by Regularization To enhance the convergence of the global aggregated model, clients update their local models by considering both the supervised learning loss and the generalization error. In addition to the conventional cross-entropy loss, FedShapleX incorporates a feature alignment mechanism to achieve this. The overall loss function can then be expressed as Eq.9.

$$loss = \mathcal{L}_{CE} + \lambda \sum_{m=1}^{M} |\mu'_m - \mu_m|_2^2 \tag{9}$$

In Eq.9, $\mathcal{L}_{CE}$ is the cross-entropy loss for supervised learning, λ is the regularization coefficient controlling the trade-off between the cross-entropy loss and the feature alignment term, while μ'_m represents the local centroid for the classification m and μ_m represents the corresponding global centroid.

Shapley Value Driven Context-Aware Submodel Extraction The FedShapleX framework is depicted in Fig.5, with its detailed steps outlined in Alg.3. For the K communication rounds, the server side manages the sub-model extraction, training, and aggregation processes while leveraging the context information to optimize sub-model extraction.

At the start of each round k, if it is not the first communication round ($k > 1$), the server will buffer the current and the last FL System status and use the experience replay to update the Deep Q Network. Then, the RL-LNS algorithm (Alg.2) is used to generate the sub-model extraction scheme S_k for the current communication round. Based on S_k, the server will extract the sub-model $\theta_{i,k}$ according to s_i from the global model. To preserve the original state of the sub-model, the server saves the unmodified parameter dictionary $w_{i,k}$ of $\theta_{i,k}$ before distributing it to the client.

Once the sub-models are distributed, each client trains the sub-model $\theta_{i,k}$ using the local data $d_{i,k}$, and returns the updated parameters $w'_{i,k}$ to the server. The server then aggregates the received local parameters $w'i, k, \forall i \in \mathcal{N}$ to obtain the global parameter W_k.

Finally, the server computes the overall individual contribution Φ_k and the class-specific individual contribution Γ_k using the global parameter W_k, the collection of unmodified local parameters $\{w_{i,k}, \forall i \in \mathcal{N}\}$, and collection of updated local parameters $\{w'_{i,k}, \forall i \in \mathcal{N}\}$. These contributions will be treated as context information for the sub-model extraction decisions in future communication rounds.

Algorithm 3 FedShapleX

Require: Dataset $\mathcal{D}$, Local Clients $\mathcal{N}$
Ensure: Global Model Θ

$\Gamma_k = \varnothing, S_k = \varnothing$
for k = 1 to K **do**
  if $k > 1$ **then**
    $\text{Buffer}(S_{k-1}, \Gamma_{k-1}, \Phi_k, S_k, \Gamma_k)$
    ExperienceReplay
  end if
  $S_k = \text{RL-LNS}(\Gamma_k, S_k)$
  for i = 1 to N **do**
    $\theta_{i,k} = \text{SplitModel}(\Theta, s_i)$
    $w_{i,k} = \text{ParameterDict}(\theta_{i,k})$
    $w'_{i,k} = \text{Distribute\&Train}(\theta_{i,k}, d_{i,k})$
  end for
  $W_k = \text{Aggregate}(\{w'_{i,k}, \forall i \in \mathcal{N}\})$
  $\Phi_k, \Gamma_k = \text{PCSV}(W_k, \{w_{i,k}, \forall i \in \mathcal{N}\}, \{w'_{i,k}, \forall i \in \mathcal{N}\})$
end for
return Θ

V. Experiment Analysis

A. Experiment Setup

Datasets and Federated Learning Scenario: To evaluate the performance of our proposed FedShapleX, we compare it against state-of-the-art (SOTA) PT-based MHFL approaches. We adopt ResNet18 as the model architecture and conduct experiments on the CIFAR-10, CIFAR-100 [21], and MNIST datasets. The experimental setup involves ten clients with varying degrees of data heterogeneity and computational capacity, simulating diverse FL scenarios. All experiments are conducted on a MacBook Pro equipped with an M2 Max chip, 64 GB RAM, and Python 3.10, utilizing the Ray and PyTorch frameworks. Due to limited computational resources, we set the number of communication rounds to 500. For the hardware heterogeneity distribution evaluation, we set the execution time to be limited to half an hour.

Baselines As we discussed earlier, FedShapleX leverages contextual information to guide the search and update of the global aggregated model. To benchmark its performance, we have compared our research with the SOTA PT-based MHFL approaches, including: *HeteroFL* [18], which uses a static scheme to extract the submodel from the global model, *Federated Dropout* [17], which randomly extracts submodel from the global model, and *FedRolex* [8], which uses a rolling scheme to extract the submodels according to the computational capacity from each layer in the global model.

Data Heterogeneity All the experiments share the same setting for the global neural network and the number of clients and have different Non-IID settings for the data distributed on different clients. We have followed FedRolex [8] to model non-IID data distributions by restricting each client to have L labels. For both CIFAR-10 and MNIST, we use $L = 2$ as the high data heterogeneity setting and $L = 5$ as the low data heterogeneity setting. As CIFAR-100 has 100 categories, we use $L = 20$ as the high data heterogeneity setting and $L = 50$ as the low data heterogeneity setting.

Computational Capacity To ensure fairness in comparing different PT-based strategies, we define five different levels of client computational capacity as $\alpha = \{1, \frac{1}{2}, \frac{1}{4}, \frac{1}{8}, \frac{1}{16}\}$. Following the definition in Eq.3, the input and output layer of the submodels remain consistent between the submodel and the global model, while the number of neurons in each hidden layer is adjusted based on the client's computational capacity.

B. Performance Comparison with SOTA PT-based Heterogeneous Federated Learning

The performance evaluation focuses on two aspects: global model accuracy, which reflects the model's overall performance on the aggregated dataset, and local accuracy, which is defined as the aggregated global model's performance on each individual distributed dataset. The latter provides a more realistic evaluation that aligns with practical FL scenarios.

Result Analysis: The result shown in the Table I. demonstrates the global and local accuracy for different datasets under different data heterogeneity settings.

Under a high data heterogeneity scenario, the FedShapleX achieves the highest global accuracy with 60.31% on the CIFAR-10 dataset, significantly outperforming all other baselines. However, when it comes to local accuracy, FedRolex achieves 59.43 %, demonstrating better performance than FedShapleX, which achieved 55.34%. This discrepancy can be attributed to FedRolex's tendency to overfit local data distributions on certain clients. Since the entire CIFAR-10 dataset is split among local clients without bias, a more generalized global model should ideally achieve both higher global accuracy and local accuracy. As local accuracy is defined as the average performance across clients, a higher local accuracy combined with a lower global accuracy suggests an imbalanced performance distribution among clients. This imbalance indicates that FedRolex or other PT-based approaches lack the ability to guide the global model updating direction toward fully utilizing the knowledge distributed to each client, resulting in a lack of generalization in the global model. This phenomenon also explains the result for HeteroFL and Federated Dropout, where they have an acceptable local accuracy but with a low global accuracy. A similar trend is observed in the MNIST dataset, where FedShapleX achieves the highest global accuracy at 76.01%, with the lowest local accuracy of 62.05%. This result is consistent with the observation on CIFAR-10. As the FedShapleX mitigates the influence introduced by Non-IID dataset distribution, it can avoid the global model's overfitting and result in an acceptable local accuracy and the best global accuracy among the baselines.

When training the FedShapleX, the RL-LNS algorithm employs a DDQN as the policy for submodel extraction. A key challenge is that the dimensionality of the input and the output embedding of the DDQN increases with the number of categories in the dataset, complicating the DDQN policy's convergence. However, FedShapleX achieves the highest global accuracy with 55.19% and the highest local accuracy of 45.06%. While it is true that a larger input and output embedding can influence the DDQN policy's convergence difficulty, our evaluation validates that our approach still guides the updated global model and outperforms state-of-the-art approaches.

Under a low data heterogeneity scenario, all experimental combinations show significant improvements. For CIFAR-10, FedShapleX achieves the highest global accuracy of 81.62%, although its local accuracy remains lower than FedRolex. However, a closer examination of FedRolex's global accuracy reveals signs of overfitting to local data distributions, which negatively impacts its global performance. For MNIST, FedShapleX also achieves the highest global accuracy of 98.95%, and the local accuracy also outperforms other methods with 55.49%. However, as MNIST is a simpler dataset than CIFAR-10, the impact of low data heterogeneity is minimal. For CIFAR-100, the improvement is less pronounced. This is because CIFAR-100 is a more complicated dataset. The larger number of classes and the more diverse feature space make it more challenging for submodels to overfit to local datasets, leading to a smaller performance gap between methods.

TABLE I
PERFORMANCE COMPARISON OF PT-BASED METHODS UNDER DIFFERENT LEVELS OF DATA HETEROGENEITY

Method	High Data Heterogeneity						Low Data Heterogeneity					
	CIFAR-10		CIFAR-100		MNIST		CIFAR-10		CIFAR-100		MNIST	
	Global	Local	Global	Local	Global	Local	Global	Local	Global	Local	Global	Local
HeteroFL	50.09	54.07	45.52	39.27	57.31	62.40	70.62	40.51	54.89	48.01	98.45	53.30
Federated Dropout	21.75	52.48	45.67	41.79	41.58	63.12	65.42	37.97	56.16	49.98	97.39	49.80
FedRolex	51.09	**59.43**	47.48	40.04	60.99	**73.32**	67.10	**58.84**	55.61	49.19	98.66	53.31
FedShapleX	**60.31**	55.34	**55.19**	**45.06**	**76.01**	62.05	**81.62**	43.51	**58.17**	**52.45**	**98.95**	**55.49**

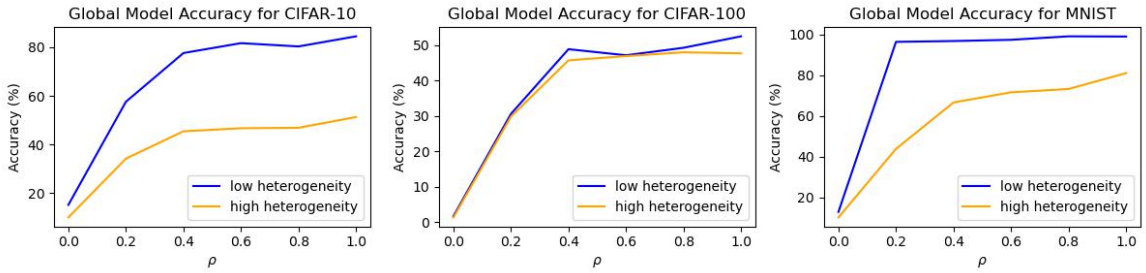

Fig. 6. Global Accuracy on CIFAR-10, CIFAR-100, and MNIST with different clients distributions

C. Hardware Heterogeneity Distribution Influence

In this experiment, we aimed to evaluate the impact of model capacity distribution across client devices. To ensure fairness, we adopted the same experiment setup as FedRolex [8]. Specifically, we utilized two client model capacities $\alpha = \{1, \frac{1}{16}\}$, representing clients with large and small computational capacities respectively. Then, we introduced ρ as the distribution ratio between clients with large and small computational capacity to investigate its effect on model performance.

Result Analysis: Fig.6 illustrates the impact of client distribution on CIFAR-10, CIFAR-100, and MNIST datasets. From the trends observed under both low heterogeneity and high heterogeneity across the three datasets, we could observe that introducing more clients with larger computational capacities improves the global model accuracy. This is because clients with higher computational capacities can simultaneously train larger submodels with more neurons involved, leading to a more balanced update on the parameters. Conversely, a FL system with more or even complete nodes having small computational capacity tends to suffer from poor performance. Such systems are more prone to overfitting, and the parameters can't be evenly trained within the system. From the perspective of the influence of the dataset, CIFAR-10 and MNIST exhibit a more significant sensitivity to heterogeneity compared with CIFAR-100. This is because CIFAR-10 and MNIST are relatively easy tasks. As a result, submodels trained on these datasets are more likely to overfit local data, amplifying the effects of data heterogeneity. In contrast, CIFAR-100, being a more complex dataset, shows better robustness towards the extent of the data heterogeneity.

VI. CONCLUSION AND FUTURE WORKS

In this work, we proposed FedShapleX, a Partially Training-based FL scheme. As state-of-the-art methods' global model updates rely entirely on aggregating updated local submodels without any guidance strategy, limiting their effectiveness in handling Non-IID data distribution. To overcome this challenge, FedShapleX introduces a Shapley value-based, context-aware submodel extraction strategy to guide the global model updates. Firstly, we proposed the Parameter Class-specific Shapley Value (PCSV) to quantify each client's contributions to each classification category. Given the vast feasible solution space for submodel extraction, we proposed the Reinforcement Learning-aided Large Neighbourhood Search (RL-LNS) algorithm to effectively explore the feasible space. FedShapleX, as a novel PT-based MHFL Scheme, uses the PCSV to assess the contribution of each client as the context and environmental information, reflecting the efficiency of the usage of knowledge distributed to clients. Then, it uses the RL-LNS algorithm to generate new sub-model extraction solutions, improving the performance of handling the Non-IID data distribution.

Our evaluation demonstrates that the FedShapleX algorithm effectively mitigates global model drift. Under scenarios with high data heterogeneity, FedShapleX outperforms the SOTA methods in global accuracy while achieving comparable or superior local accuracy. In future work, we will further investigate the impact of the sub-model extraction policy on convergence and explore novel training techniques to improve policy efficiency. Additionally, we will explore some advanced feature embedding methods to enhance the scalability and efficiency of FedShapleX in a FL system with a larger number of clients and datasets containing more categories.

REFERENCES

[1] S. Banabilah, M. Aloqaily, E. Alsayed, N. Malik, and Y. Jararweh, "Federated learning review: Fundamentals, enabling technologies, and future applications," *Information processing & management*, vol. 59, no. 6, p. 103061, 2022.

[2] Y. Zhang, D. Zeng, J. Luo, X. Fu, G. Chen, Z. Xu, and I. King, "A survey of trustworthy federated learning: Issues, solutions, and challenges," *ACM Transactions on Intelligent Systems and Technology*, vol. 15, no. 6, pp. 1–47, 2024.

[3] L. Daliang and W. Junpu, "Fedmd: Heterogenous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, vol. 2, 2019.

[4] S. Wang, Y. Fu, X. Li, Y. Lan, M. Gao *et al.*, "Dfrd: Data-free robustness distillation for heterogeneous federated learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.

[5] L. Yi, H. Yu, G. Wang, and X. Liu, "Fedlora: Model-heterogeneous personalized federated learning with lora tuning," *arXiv preprint arXiv:2310.13283*, 2023.

[6] T. Shen, J. Zhang, X. Jia, F. Zhang, G. Huang, P. Zhou, K. Kuang, F. Wu, and C. Wu, "Federated mutual learning," *arXiv preprint arXiv:2006.16765*, 2020.

[7] 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," *Advances in Neural Information Processing Systems*, vol. 34, pp. 12 876–12 889, 2021.

[8] S. Alam, L. Liu, M. Yan, and M. Zhang, "Fedrolex: Model-heterogeneous federated learning with rolling sub-model extraction," *Advances in neural information processing systems*, vol. 35, pp. 29 677–29 690, 2022.

[9] H. Zhu, J. Xu, S. Liu, and Y. Jin, "Federated learning on non-iid data: A survey," *Neurocomputing*, vol. 465, pp. 371–390, 2021.

[10] T. Wang, J. Rausch, C. Zhang, R. Jia, and D. Song, "A principled approach to data valuation for federated learning," *Federated Learning: Privacy and Incentive*, pp. 153–167, 2020.

[11] Z. Fan, H. Fang, Z. Zhou, J. Pei, M. P. Friedlander, C. Liu, and Y. Zhang, "Improving fairness for data valuation in horizontal federated learning," in *2022 IEEE 38th International Conference on Data Engineering (ICDE)*. IEEE, 2022, pp. 2440–2453.

[12] Z. Fan, H. Fang, Z. Zhou, J. Pei, M. P. Friedlander, and Y. Zhang, "Fair and efficient contribution valuation for vertical federated learning," *arXiv preprint arXiv:2201.02658*, 2022.

[13] Z. Liu, Y. Chen, H. Yu, Y. Liu, and L. Cui, "Gtg-shapley: Efficient and accurate participant contribution evaluation in federated learning," *ACM Transactions on intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–21, 2022.

[14] X. Xu, L. Lyu, X. Ma, C. Miao, C. S. Foo, and B. K. H. Low, "Gradient driven rewards to guarantee fairness in collaborative machine learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 16 104–16 117, 2021.

[15] N. Tastan, S. Fares, T. Aremu, S. Horvath, and K. Nandakumar, "Redefining contributions: Shapley-driven federated learning," *arXiv preprint arXiv:2406.00569*, 2024.

[16] T. Shen, J. Zhang, X. Jia, F. Zhang, G. Huang, P. Zhou, K. Kuang, F. Wu, and C. Wu, "Federated mutual learning," 2020. [Online]. Available: https://arxiv.org/abs/2006.16765

[17] D. Wen, K.-J. Jeon, and K. Huang, "Federated dropout—a simple approach for enabling federated learning on resource constrained devices," *IEEE wireless communications letters*, vol. 11, no. 5, pp. 923–927, 2022.

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

[19] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *arXiv preprint arXiv:1602.05629*, vol. 2, no. 2, 2016.

[20] H. Van Hasselt, A. Guez, and D. Silver, "Deep reinforcement learning with double q-learning," in *Proceedings of the AAAI conference on artificial intelligence*, vol. 30, no. 1, 2016.

[21] A. Krizhevsky, G. Hinton *et al.*, "Learning multiple layers of features from tiny images," 2009.