Communication-Efficient Federated Learning for Heterogeneous Clients

YING LI, School of Computer Science and Engineering, Northeastern University College of Computer Science and Engineering, Shenyang, China and TU Wien, Vienna, Austria
XINGWEI WANG*, Northeastern University College of Computer Science and Engineering, Shenyang, China
HAODONG LI, Northeastern University College of Computer Science and Engineering, Shenyang, China
PRAVEEN KUMAR DONTA, Stockholm University, Stockholm, Sweden
MIN HUANG, Northeastern University, Shenyang, China
SCHAHRAM DUSTDAR, Distributed Systems Group, TU Wien, Vienna, Austria

Federated learning stands out as a promising approach within the domain of edge computing, providing a framework for collaborative training on distributed datasets without necessitating data sharing. However, federated learning involves the frequent transmission of machine learning model updates between the server and clients, resulting in high communication costs. Additionally, heterogeneous clients can further complicate the Federated Learning process and deteriorate performance. To address these challenges, we propose *adaptive self-knowledge distillation-based quality- and reputation-aware cross-device federated learning* (ASDQR) - an efficient communication and inference framework designed for heterogeneous clients. ASDQR initiates the process by selecting high-reputation and high-quality clients to be involved in federated learning, significantly impacting communication efficiency and inference effectiveness. ASDQR also introduces a model of adaptive local self-knowledge distillation that incorporates multiple local personalized historical knowledge for more accurate inference, allowing the historical level to be dynamically adjusted across time. Finally, we present an inference-effective aggregation scheme that assigns higher weights to important and reliable local model updates based on clients' contribution degrees when performing global model aggregation. ASDQR consistently outperforms baseline methods across all datasets and communication rounds, achieving 9.0% higher accuracy than FedAvg, 6.59% higher than MOON, 0.29% higher than FedProx, 0.2% higher than PFedSD, and 0.08% higher than FedMD on the MNIST dataset at 100 communication rounds. Similar improvements are observed on CIFAR, HAR, and WISDM datasets, demonstrating the robustness and efficiency of ASDQR in federated learning with non-IID data.

CCS Concepts: • **Networks** → **Communication Networks**; • **Computing** → **AI**.

Additional Key Words and Phrases: Federated Learning, Data Heterogeneity, Knowledge Distillation, Communication Efficiency, Inference Effectiveness.

*Corresponding author: Xingwei Wang.

Authors' Contact Information: Ying Li, School of Computer Science and Engineering, Northeastern University College of Computer Science and Engineering, Shenyang, Liaoning, China and TU Wien, Vienna, Vienna, Austria; e-mail: liying1771@163.com; Xingwei Wang, Northeastern University College of Computer Science and Engineering, Shenyang, Liaoning, China; e-mail: wangxw@mail.neu.edu.cn; HaoDong Li, Northeastern University College of Computer Science and Engineering, Shenyang, Liaoning, China; e-mail: ihaodong0811@163.com; Praveen Kumar Donta, Stockholm University, Stockholm, Stockholm, Sweden; e-mail: praveen@dsv.su.se; Min Huang, Northeastern University, Shenyang, Liaoning, China; e-mail: mhuang@mail.neu.edu.cn; Schahram Dustdar, Distributed Systems Group, TU Wien, Vienna, Vienna, Austria; e-mail: dustdar@dsg.tuwien.ac.at.

ACM 1557-6051/2025/2-ART
https://doi.org/10.1145/3716870

1 Introduction

With the advent of Deep Learning (DL) and the popularity of the Internet of Things (IoT) in recent years, edge devices generate massive amounts of data. Multi-access Edge Computing (MEC) [5], regarded as a promising technique for 5G heterogeneous networks, allows edge devices to access computing resources effectively. This is especially attractive for a wide range of IoT applications. As a consequence of industry competition and increasingly stringent data protection regulations [39], private data sharing between different clients or platforms has been severely restricted. However, the achievement of high-quality DL models within the MEC architecture necessitates an abundance of available shared data. As an emerging privacy-preserving distributed learning paradigm in Artificial Intelligence (AI), Federated Learning (FL) [26] is a promising technology to address the challenge above. The integration of MEC and FL enables edge nodes to cooperatively train a global model by sending local model updates derived from individual raw data to the cloud server. This method preserves data utility integrity while safeguarding data privacy.

Despite the aforementioned advantages of FL, its implementation faces several challenges. Firstly, the proliferation of edge devices and the escalating complexity of models deployed on these devices present a notable hurdle. Concurrently, neural network models employed in DL typically comprise millions or even billions of parameters [4]. In contrast, the network bandwidth between edge devices and cloud servers is constrained, with uplink speeds considerably slower than downlink speeds, resulting in elevated communication costs and delays. Moreover, the clients and data involved in FL model training may exhibit heterogeneity and unreliability, leading to instances where not all locally trained models contribute positively to overall model convergence. If unexpected models are sent to the cloud server for aggregation, it will not only deteriorate the performance of the trained model but also escalate additional communication costs. Lastly, communication costs between the cloud server and clients may account for over 70% of the total energy consumption. Consequently, optimizing the use of communication resources to reduce bandwidth requirements, training time, and energy consumption while maintaining high performance in MEC environments is a critical research focus.

While various attempts have been made to explore research on communication-efficient FL in the literature, this study remains essential for addressing their challenges. Techniques such as model compression, including parameter pruning [31], sparsification [35], and quantization [3], reduce the size of updates but often compromise model accuracy or introduce additional computational burdens. Increasing the number of local training iterations is another common approach [32]; while this reduces communication frequency, it can lead to model drift, especially in scenarios where client data is Non-Independent and Identically Distributed (non-IID). Furthermore, selecting a subset of clients for participation in training rounds is a practical way to lower communication overhead, but methods like random client selection [22, 32] fail to optimize model performance, and advanced client selection techniques based on client contributions or reinforcement learning [30, 46] often require substantial computational resources. Other promising approaches include federated transfer learning, which adapts pre-trained models to local datasets, and adaptive compression techniques, which dynamically adjust communication based on client capabilities. While these methods provide notable benefits, they are limited by their reliance on pre-trained models, computational complexity, or inability to adapt effectively to heterogeneous client environments [7, 36]. Despite these advancements, existing methods often involve trade-offs between communication efficiency, model accuracy, and scalability, particularly when dealing with non-IID data and resource-constrained clients. Addressing these challenges requires a more adaptive and holistic framework that balances efficiency and performance across diverse scenarios. The following research question has motivated us to address the above challenges described: How can we design an adaptive, efficient, and effective FL framework for heterogeneous clients that optimizes communication bandwidth utilization and enhances FL performance in MEC environments? Succinctly, our approach involves selecting clients that positively influence model training on non-IID data and aggregating local model updates through adaptive self-knowledge distillation.

This paper's primary contributions are delineated as follows:

- Novel FL Framework: To tackle the challenges of communication costs and inference inefficiency in MEC environments, we propose Adaptive Self-Knowledge Distillation-Based Quality-Aware and Reputation-Aware Cross-Silo FL (ASDQR). This innovative framework integrates client selection, adaptive self-knowledge distillation, and inference-effective aggregation into a unified system, ensuring both communication efficiency and inference effectiveness.
- Quality-Aware and Reputation-Aware Client Selection: ASDQR introduces a client selection mechanism that evaluates edge nodes based on data quality, scale, label distribution, distribution differences, and client reputation. By incorporating the freshness of learning records, it dynamically prioritizes high-contribution clients to drive global model updates. This approach enhances communication efficiency by minimizing redundant communication and maximizing the impact of each communication round, particularly in heterogeneous environments.
- Adaptive Self-Knowledge Distillation with Historical Knowledge: We propose an adaptive self-knowledge distillation scheme that utilizes historical local knowledge from prior models to guide current training. By dynamically assembling a personalized ensemble of historical model updates, this approach improves inference precision and generalization in non-IID scenarios. By enhancing local model quality, it reduces redundant or low-quality updates, indirectly decreasing communication costs.
- Inference-Effective Aggregation Scheme: The proposed framework optimizes global model aggregation by assigning higher weights to clients with greater contributions, as determined by their historical performance, the quality of local model updates, and reliability metrics. This weighted aggregation improves global model accuracy and reduces communication rounds, ensuring robustness and efficiency in heterogeneous environments.
- Experiments: Extensive experiments on real-world datasets demonstrate that ASDQR consistently outperforms baseline methods across various settings and communication rounds. On the MNIST dataset, ASDQR achieves a 9.0% accuracy improvement over FedAvg, 6.59% over MOON, 0.29% over FedProx, 0.20% over PFedSD, and 0.08% over FedMD on CNN at 100 communication rounds. Similar accuracy gains are observed across CIFAR, HAR, and WISDM datasets, highlighting ASDQR's robustness and scalability for FL tasks in non-IID scenarios.

The rest of the paper is arranged as follows. In Section 2, we perform literature reviews on the most recent and related works. In Section 3, we describe the system model and formulate our problem. In Section 4, we describe the proposed work. In Section 5, we evaluate our work in various scenarios and compare it with the literature. Finally, we conclude our work with a future scope in Section 6.

2 Related Work

In this section, we present the related work regarding communication-efficient FL and inference-effective FL, and Self-Knowledge Distillation in FL.

Communication-Efficient FL. Communication cost is a significant challenge in FL, limiting its practical adoption. Numerous strategies have been proposed to enhance communication efficiency in FL, which can be broadly categorized into five primary approaches: 1) Model Compression: This method reduces the size of models exchanged between clients and the central server. Itahara *et al.* [16] achieved reductions in communication and computation costs through neural network pruning. Sattler *et al.* introduced STC [35], leveraging parameter sparsification for efficient weight update compression. Bernstein *et al.* [3] proposed SIGNSGD, which minimizes communication overhead using model parameter quantization. Wu *et al.* developed FedKD [43], combining adaptive mutual knowledge distillation and dynamic gradient compression to enhance both communication

efficiency and effectiveness. While these techniques reduce communication costs, they often increase computational overhead on resource-constrained devices and may compromise model accuracy in scenarios involving heterogeneous data distributions. 2) Local Updates: This approach allows clients to train locally on their datasets and only transmit model updates to the central server. McMahan *et al.* [32] reduced the frequency of parameter exchanges by performing multiple local updates before communication. However, prolonged local updates can lead to model drift in non-IID settings, which may cause suboptimal global model convergence and diminished performance in diverse client environments. 3) Client Selection: This strategy minimizes communication overhead by selecting a subset of clients for participation. Random selection [22, 32, 41] is widely used but may degrade model performance. Contribution- and performance-based selection [8, 30, 45] focuses on choosing clients based on their impact on model accuracy and convergence time, using metrics like data quality and contribution. Reinforcement learning-based approaches [10, 46, 47] adaptively select clients based on metrics such as energy consumption and training delay. System optimization techniques [1, 15, 44, 48] further improve efficiency by considering resource availability and environmental factors. Despite their benefits, these methods often require significant computational resources and struggle to adapt dynamically to changing client conditions. Detailed comparisons can be found in Table 3 of Appendix A. 4) Federated Transfer Learning: This technique involves transferring a pre-trained model to clients, which is then fine-tuned using their local data. Sharma *et al.* [36] significantly improved runtime and communication efficiency through secure federated transfer learning. Cheng *et al.* [7] combined federated transfer learning with reinforcement learning-based client selection, achieving notable gains in model accuracy and communication efficiency. Their reliance on pre-trained models, however, limits applicability in scenarios with diverse clients, where global pre-training may not align with local data distributions. 5) Adaptive Compression: This method dynamically selects the most suitable compression algorithm based on data characteristics. Hönig *et al.* [13] proposed DAdaQuant, which adjusts quantization levels dynamically across clients and time. Wang *et al.* [41] introduced FedCAMS, an adaptive optimization approach that mitigates communication overhead. Diao *et al.* [11] developed HeteroFL, which allocates submodels to devices based on their computational and communication capabilities, optimizing both communication and computation. While these methods enhance communication efficiency, their computational complexity and reliance on real-time adjustments may limit adoption in resource-constrained environments.

Inference-Effective FL. Enhancing inference effectiveness in FL is challenging due to the decentralized and heterogeneous nature of the data and clients. Several approaches have been proposed to address these challenges: 1) Client Selection: This strategy involves selecting a subset of clients to perform inference tasks, thereby improving effectiveness and minimizing overall inference time. The efficacy of FL models is influenced by the quality of data and computational resources available to the selected clients. Methods such as ClusterFL and PyramidFL [23, 33] prioritize clients with high-quality data and computational power to enhance inference performance. However, many client selection approaches rely on static criteria, making them less effective in adapting to dynamic client conditions. 2) Adaptive Federated Optimization: This method dynamically adjusts optimization algorithms based on client characteristics and data distribution, enhancing both model effectiveness and inference accuracy. Jin *et al.* proposed pFedSD [18], which improves robustness and effectiveness through personalized optimization. Jayaram *et al.* [17] introduced AdaFed, leveraging serverless/cloud functions for efficient and fault-tolerant adaptive aggregation in FL. Despite these advancements, many federated optimization techniques fail to fully incorporate personalized training, leading to reduced inference accuracy in non-IID scenarios.

Self-Knowledge Distillation in FL. Self-knowledge distillation in FL enhances model optimization and personalization by enabling models to adapt to heterogeneous data distributions without external teacher models, thereby reducing communication overhead and improving performance across diverse client datasets. Jin *et al.*[18] introduced pFedSD, an innovative personalized FL framework employing self-knowledge distillation to mitigate historical knowledge forgetting and enhance model performance through an implicit local model ensemble.

Singh *et al.*[37] developed PerFed-SKD, a personalized FL framework utilizing self-knowledge distillation to streamline knowledge transfer across model iterations, easing training on resource-constrained devices while harmonizing generalization and personalization. He *et al.*[12] introduced FedCAD, a class-wise adaptive self-distillation approach for FL, which assessed the global model's inference confidence across different categories and dynamically adjusts its influence on local training, effectively mitigating the adverse effects of non-IID data distribution. Wang *et al.*[38] introduced a backbone self-distillation approach for personalized FL, enabling clients to train models with unique weights for global updates and personalization, merging universal representation with bespoke customization. Existing Self-Knowledge Distillation approaches in FL primarily focus on short-term optimization, static integration of knowledge, and limited adaptability to non-IID challenges and resource constraints.

Despite the significant advancements in communication efficiency, inference effectiveness, and self-knowledge distillation in FL, several research gaps remain unaddressed. Existing works on communication efficiency often introduce trade-offs, such as increased computational overhead or compromised model accuracy, especially in heterogeneous and resource-constrained environments. Similarly, while inference optimization approaches like client selection and adaptive federated optimization improve model performance, they often rely on static criteria or fail to incorporate personalized optimization effectively, limiting adaptability to non-IID scenarios. Furthermore, self-knowledge distillation methods focus primarily on short-term optimization or static integration of knowledge, lacking the flexibility to dynamically adjust to evolving client conditions and heterogeneous data distributions. To address these gaps, this work seeks to reduce communication costs by prioritizing high-contribution clients. To guide the selection process, we evaluate both current and historical quality and reputation metrics, considering factors such as data quality, data scale, data heterogeneity, and client reputation, while accounting for the freshness of learning records. Additionally, we aim to optimize inference effectiveness and minimize communication costs by integrating client selection, adaptive personalized optimization, and inference-effective aggregation into a unified framework. By bridging these gaps, our framework offers a scalable and adaptable solution tailored for real-world heterogeneous FL environments.

3 System Description and Problem Definition

In this section, we initially introduce the cloud-edge-based cross-silo FL system. Subsequently, we provide a formalization of the problem studied in this paper.

3.1 System Description

In this work, we focus on the cloud-edge-based cross-silo FL system, as illustrated in Figure 1. In the FL system, our scenario considers a cloud server and N edge servers (each edge server represents an organization), and each edge server covers all the edge devices in that area. Let $\mathcal{N} = \{1, 2, \cdots, n, \cdots, N\}$ represents the set of edge servers. For each edge server n, where $n \in \mathcal{N}$, there exists a local dataset $\mathcal{S}_n$ collected from the edge devices it covers. The size of this local dataset is represented by $S_n = |\mathcal{S}_n|$. The edge server n is capable of selecting a subset $\mathcal{X}_n \subseteq \mathcal{S}_n$ from its local dataset for local model training. Here, x_n represents the size of the chosen subset, i.e., $x_n = |\mathcal{X}_n|$. The frequently used symbols in this paper are meticulously detailed in Table 1. The detailed procedure for Cloud-edge-based cross-silo FL training is elaborated below.

Step 1: Download. In the initial phase, the edge servers are required to download the global model aggregated from the cloud server in the preceding round t (Note that the global model is randomly initialized during the first round).

$$\omega_n^t = \omega^t. \tag{1}$$

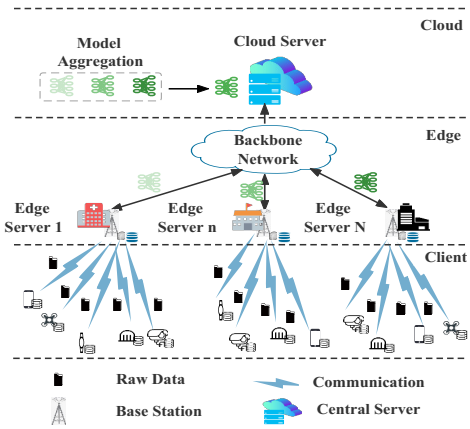


Fig. 1. The overview of cloud-edge based cross-silo FL.

Step 2: Updates. Edge servers proceed to train their local models by utilizing their respective local raw data in accordance with the downloaded global model.

$$\omega_n^{t+1} = \omega_n^t - \eta \nabla L_n(\omega_n^t; x_n^t). \tag{2}$$

Step 3: Upload. Edge servers transmit their local model updates to the central server.

Step 4: Aggregation. Cloud server aggregates all selected local model updates to generate a new global model, where $\mathcal{N}_s$ is the clients selected to participate in the model training.

$$\omega^{t+1} = \frac{\sum_{n \in \mathcal{N}_s} [x_n \omega_n^{t+1}]}{\sum_{n \in \mathcal{N}_s} x_n} \tag{3}$$

Multiple rounds are necessary to achieve convergence in FL, with each round comprising K iterations. The objective of FL is to obtain the optimal weights, denoted as ω^*, for the global model to minimize the global loss function $L(\omega)$.

$$L(\omega) = \sum_{n \in \mathcal{N}_s} \frac{x_n}{\sum_{n \in \mathcal{N}_s} x^n} L_n(\omega_n; \mathcal{X}_n). \tag{4}$$

where $L_n(\omega_n; \mathcal{X}_n)$ represents the loss for subset $\mathcal{X}_n$ with respect to ω.

$$\omega^* = \arg\min_{\omega} L(\omega). \tag{5}$$

3.2 Problem Definition

The communication efficiency of FL is determined by the time required to upload model updates from edge clients to the cloud server and the total volume of data (in bits) transferred between edge clients and the cloud.

Communication Time: The communication time in FL consists of two components: the download time for global model updates from the cloud server to edge servers (*downlink*) in Step 1 and the upload time for local model updates from edge servers to the cloud server (*uplink*) in Step 3. Among these, the uplink bandwidth often represents a more significant bottleneck due to several reasons. First, internet providers generally prioritize download speeds over upload speeds to enhance user experiences for activities like streaming and file downloading. Second, in FL, the downlink operation involves broadcasting the same global model to all clients, which is typically more efficient compared to aggregating diverse parameters from multiple clients [13]. As such, this work focuses on uplink communication time while neglecting the comparatively minor contribution of downlink time.

Given the constraints of limited network bandwidth and the increasing number of connected clients, the data transfer rate decreases as more clients are added. According to Shannon's theorem, the maximum data transfer rate for a communication link is given by:

$$C = B \times log_2 \left(1 + \frac{SNR}{N_s}\right) \tag{6}$$

where C is the channel's maximum capacity, B denotes the channel bandwidth, SNR represents the signal-to-noise ratio, and N_s is the number of clients participating in FL. Assuming that SNR remains constant throughout the FL process and that data is transmitted independently from each edge server to the central server, an increase in N_s reduces the channel capacity C due to bandwidth sharing among clients. This reduction can adversely affect the aggregation of complex models requiring large amounts of data. Thus, the communication time for one round of FL is defined as:

$$T_{comm} = \frac{S_m}{C} = \frac{S_m}{B \times log_2 \left(1 + \frac{SNR}{N_s}\right)} \tag{7}$$

Here, S_m represents the total size of the uploaded model parameters from all selected edge servers in a round. From Eq. (7), it is evident that T_{comm} increases with the number of selected clients N_s. Moreover, T_{comm} is directly proportional to S_m, indicating that an increase in the size of the uploaded model updates also leads to higher communication time. This, in turn, impacts the overall training time of the system.

To achieve communication-efficient and inference-effective FL based on Eq. (6) and (7), we employ two key strategies: 1) Quality-Aware and Reputation-Aware Client Selection: Instead of random client selection, the proposed mechanism prioritizes clients with higher contributions to model training. By selecting a subset of clients that provide the most valuable updates, the total number of participating clients (N_s) is reduced, maximizing the utility of the available communication bandwidth. 2) Model Update Compression: Self-distillation offers an effective approach to reduce communication time (T_{comm}) by optimizing model update compression, thereby decreasing the size of transmitted updates (S_m) while preserving their quality. These strategies work in tandem to optimize the use of communication resources while maintaining high model accuracy and robustness in non-IID and heterogeneous scenarios.

3.3 Problem Difficulty

Formulating an effective and efficient FL strategy poses significant challenges for several reasons. 1) Data Heterogeneity: Numerous research assumes that the local data distribution is homogeneous and the local data samples are IID across all clients, which is often inconsistent with real-world scenarios. Data heterogeneity, in reality, can increase communication requirements as clients may need to exchange more information to ensure comprehensive model training. 2) Model Reinitialization and Catastrophic Forgetting: Each communication round in FL starts by initializing the local model with the latest global model. This initialization process discards previously acquired local knowledge, necessitating a restart in training the local model for each new communication round. Such a process often leads to catastrophic forgetting, significantly degrading the local model's performance, especially in non-IID settings. 3) Aggregate Oversight: The FedAvg algorithm considers only the quantity of local data samples contributed by each client for global model aggregation, neglecting the importance and credibility of local model updates. Consequently, the global model might undervalue or overlook clients whose local data is exceptionally valuable and credible, thereby impacting overall performance. Additionally, FedAvg's assumption that local data samples are IID across all clients can lead to biased local model updates and suboptimal model performance. Our proposed framework is designed for communication efficiency and effective inference, systematically addressing these issues.

Table 1. Frequently Used Symbols.

Symbol	Description	Symbol	Description
$\mathcal{N}$	A set of edge servers.	N	A number of edge servers.
$\mathcal{N}_s$	A set of selected edge servers.	N_s	A number of selected edge servers.
$\mathcal{S}_n$	Local dataset of EN n.	S_n	Size of $\mathcal{S}_n$.
T_{comm}	The communication time of FL in one round.	$\mathcal{X}_n$	Chosen subset for EN n.
x_n	Size of $\mathcal{X}_n$.	ω_n	Client n's local model.
ω_n^t	Local model of client n at round t.	ω^{t+1}	Global model at round $t+1$.
$L_n(\omega;\mathcal{X}_n)$	Local loss function of client n.	$L(\omega)$	The global loss function.
h_n^t	The label distribution of client n.	kl_n^t	The distribution difference of client n.
K	Number of iterations in one round.	B	Channel bandwidth.
SNR	Signal-to-noise ratio.	C	Maximum speed supported by the channel.
q_n^t	Learning quality of EN n in round t.	$loss_n^t$	Training loss of client n in round t.
$loss_{avg}^t$	Average training loss of all clients in round t.	α_n	Number of positive interactions.
β_n	Number of negative interactions.	q_n	A set of learning quality.
R_n	A set of learning reputation.	p_n	Success probability of packet transmission.
S_m	Total size of the uploaded model updates in one round.	$S_n^{m(t)}$	Number of bits uploaded from the edge servers n in round t.

4 Design Details

In this section, we delineate ASDQR's three major components: 1) Client Selection with quality and reputation awareness; 2) Adaptive local model update with historical knowledge; and 3) Inference effective aggregation scheme. Figure 2 provides an overview of the ASDQR framework.

4.1 Client Selection with Quality and Reputation-Awareness

In this component, we assume that the round t commences at I_{t-1}^t and concludes at I_t^{t+1}. Clients desiring to participate in training must submit their model updates by I_n^t within the interval $[I_{t-1}^t, I_t^{t+1}]$, failing which they will be excluded from round t. During round t, only chosen clients are eligible to engage in model aggregation at I_t^{t+1}. Subsequently, the succeeding round starts.

4.1.1 Quality Estimation. The quality estimation process comprises two distinct stages: learning quality quantification and the current learning quality estimation for n^{th} client with interaction freshness, elucidated in the subsequent discussion.

a) Learning Quality Quantification: A cross-entropy divergence measurement can be used to evaluate client impact in globally trained models, as suggested by Chen *et al.* [6]. In any case, this approach introduces considerable overhead during data transmission. An alternative approach, proposed by Deng *et al.* [9] emphasizes the disparity between mean loss of global task and local model average training loss. The loss disparity n^{th} client at round t is in Eq.(8):

$$l_n^t = loss_{avg}^t - loss_n^t. \tag{8}$$

where the training loss of client n be denoted as $loss_n^t$ in round t, and the mean loss across all clients in round t be represented as $loss_{avg}^t$.

In assessing the quality of clients within FL, it is recognized that relying solely on loss values may offer an incomplete depiction of data quality. Hence, we advocate for the development of a composite scoring function that considers multiple factors, encompassing loss values, data volume, and label distribution. This composite scoring function serves as a robust metric for evaluating and ranking the quality of client data. The proposed

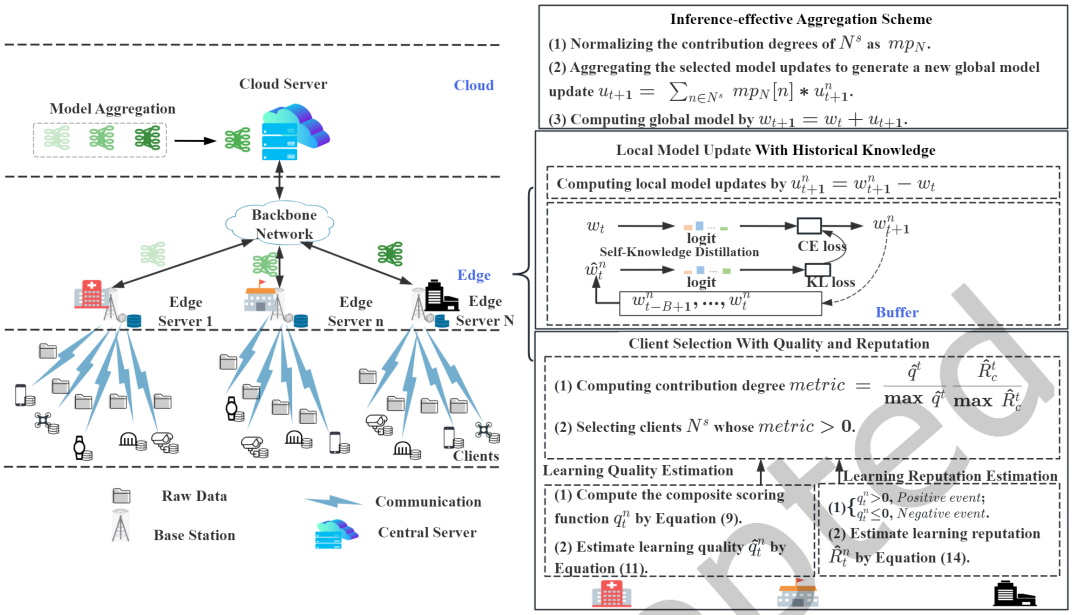


Fig. 2. Overview of proposed ASDQR.

composite scoring function of learning quality is articulated as follows:

$$q^t_n = \begin{cases} w_1(l^t_n)' + w_2(x^t_n)' + w_3(h^t_n)' + w_4(kl^t_n)', & \text{if } I^t_{t-1} < I^t_n \leq I^{t+1}_t; \\ 0, & \text{Otherwise.} \end{cases} \tag{9}$$

Given that $w_1 + w_2 + w_3 + w_4 = 1$, where l^t_n, x^t_n, h^t_n, and kl^t_n represent the loss disparity, data volume, and label distribution, and distribution difference, respectively. We implemented a weight optimization algorithm designed to maximize a performance metric by adjusting the weights assigned to different data categories, including loss, volume, label, and distribution differences. The algorithm optimizes these weights by solving a constrained minimization problem using the SLSQP method, ensuring that the weights sum to 1 and remain within the range [0, 1]. The corresponding normalized values are denoted as $(l^t_n)'$, $(x^t_n)'$, $(h^t_n)'$ and $(kl^t_n)'$. Specifically, h^t_n is defined as Eq. (10):

$$h^t_n = \frac{100 \times H^t_{min}}{H^t_{max}} \tag{10}$$

where H^t denotes the list of the occurrence frequencies for each label in round t and Eq. (10) is formulated to quantify the extent of imbalance in the frequency of label occurrences. A larger disparity between H^t_{max} and H^t_{min} corresponds to an increased h^t_n, indicative of more pronounced label distribution discrepancies. The normalization of the percentage values facilitates a standardized metric for comparing the gap, encompassing a uniform range from 0 to 100.

The KL divergence, denoted as kl^t_n, is used to measure the difference between the global and local data distributions, as defined in Eq. (11):

$$kl_n = \sum_{c=1}^{C} P_{\text{client }_n}(c) \cdot \log \frac{P_{\text{client }_n}(c)}{P_{\text{global}}(c)} \tag{11}$$

Algorithm 1 Quality and Reputation Estimation (QRE).

Input: Average loss $loss^{avg}$, the list of the label frequencies in round t H^t, the global distribution of class c $P^{t_c}_{\text{global}}(c)$, the local distribution of class c for client n $P^{t_c}_{\text{client}_n}(c)$, # positive/negative interactions α_n/β_n, learning quality/reputation set q_n/R_n, packet transmission success probability p_n.

Output: Quality estimation q_{t_c}, reputation estimation R_{t_c}.

Initialize $q_{t_c} \leftarrow \emptyset$, $R_{t_c} \leftarrow \emptyset$
for each client $n \in \mathcal{N}_s$ in parallel **do**
  $l^{t_c}_n \leftarrow loss^{t_c}_{avg} - loss^{t_c}_n$
  $h^{t_c}_n = \frac{100 \times H^t_{min}}{H^t_{max}}$
  $kl^{t_c}_n = \sum_{c=1}^{C} P^{t_c}_{\text{client}_n}(c) \cdot \log \frac{P^{t_c}_{\text{client}_n}(c)}{P^{t_c}_{\text{global}}(c)}$
  Nomalize $l^{t_c}_n, h^{t_c}_n, x^{t_c}_n, kl^{t_c}_n$ to $(l^{t_c}_n)', (h^{t_c}_n)', (x^{t_c}_n)', (kl^{t_c}_n)'$
  $q^{t_c}_n \leftarrow w_1(l^{t_c}_n)' + w_2(x^{t_c}_n)' + w_3(h^{t_c}_n)' + w_4(kl^{t_c}_n)'$
  Update $q_n \leftarrow q_n + \{q^{t_c}_n\}$
  Estimate learning quality $\hat{q}^{t_c}_n$ ▷ Eq. (12)
  Update $q_{t_c} \leftarrow q_{t_c} + \{\hat{q}^{t_c}_n\}$
  if $\hat{q}^{t_c}_n > 0$ **then**
    Update $\alpha_n \leftarrow \alpha_n + 1$
  else
    Update $\beta_n \leftarrow \beta_n + 1$
  end if
  $b^t_n \leftarrow p_n \frac{\gamma\alpha_n}{\gamma\alpha_n + \delta\beta_n}$
  $u_n \leftarrow 1 - p_n$
  $R^t_n \leftarrow b^t_n + au_n$
  Update $R_n \leftarrow R_n + \{R^{t_c}_n\}$
  Estimate learning reputation $\hat{R}^{t_c}_n$ ▷ Eq. (15)
  Update $R_{t_c} \leftarrow R_{t_c} + \{\hat{R}^{t_c}_n\}$
end for
return q_{t_c}, R_{t_c}

Here, $P_{\text{global}}(c)$ represents the global distribution of class c, while $P_{\text{client}_n}(c)$ denotes the distribution of class c for client n. A global data distribution is calculated by normalizing the frequency of each class over the entire dataset, while a client data distribution is determined by normalizing a subset of data.

b) The Current Learning Quality estimation of client n with Interaction Freshness: The n-th client's data undergoes changes in every round. When a malicious node contributes high-quality data in a single round, it can engage in malicious activities during model training and exploit the training opportunity. Contrary to cross-device FL, cross-silo FL possesses an identity and allows clients to carry their state from round to round. Consequently, based on the historical learning quality records of client n, we can estimate its current learning quality. We employ a freshness fading function to incorporate time effects on learning quality through assigning weights: $\zeta(t) = \rho^{t_{c-1}-t}$, where $\rho \in (0, 1)$ represents a given fade parameter concerning quality freshness, t_c is the most recent round, and t is the training round within $[t_0, t_1, t_2, \ldots, t_{c-1}]$ [19]. Here, we determined the optimal fade parameter ρ by conducting a grid search over the range [0.01, 0.99], selecting the value that maximized our learning quality metrics in estimating the learning quality's freshness and relevance over time. This optimization

helps understand how different ρ values influence the model's capacity to represent the dynamic nature of learning quality over time, acknowledging that ρ itself might vary with each round and across time. Thus, the current learning quality of client n up to round t is computed by Eq. (12).

$$\hat{q}_n^{t_c} = \frac{\sum_{t=t_0}^{t_{c-1}} [\zeta(t) q_n^t]}{\sum_{t=t_0}^{t_{c-1}} \zeta(t)}. \tag{12}$$

4.1.2 Reputation Estimation. Similarly to quality estimation, reputation estimation consists of two distinct stages: learning reputation quantification and current learning reputation estimation of client n with fresh interactions.

a) Learning Reputation Quantification: In this study, we utilize a subjective logic model [29], an extensively employed probabilistic reasoning framework, to derive the learning reputation of client n based on its learning quality.

$$\begin{aligned} b_n^t &= p_n^t \frac{\gamma \alpha_n^t}{\gamma \alpha_n^t + \delta \beta_n^t} \\ d_n^t &= p_n^t \frac{\delta \beta_n^t}{\gamma \alpha_n^t + \delta \beta_n^t} \\ u_n^t &= 1 - p_n^t. \end{aligned} \tag{13}$$

In the proposed FL framework, the reputation opinion of client n is represented by a tuple vector $\gamma_n^t = \{b_n, d_n, u_n\}$ in round t. Here, b_n, d_n, and u_n denote belief, disbelief, and uncertainty, respectively. The constraints are $b_n + d_n + u_n = 1$, with $b_n, d_n, u_n \in [0, 1]$. Additionally, α_n^t and β_n^t represent the numbers of positive and negative interactions in the round t separately. The cloud server regards the training process as a positive interaction if the learning quality of the client n is > 0, and vice versa. Furthermore, p_n^t signifies the successful packet transmission probability, influencing the uncertainty of the opinion. Consequently, the client n's reputation in round t is formulated as Eq. (14).

$$R_n^t = b_n^t + a u_n^t. \tag{14}$$

where the coefficient $a \in [0, 1]$ delineates the extent to which uncertainty influences reputation.

In order to mitigate the impact of negative interaction events, a greater weight is allocated to negative interactions during the reputation calculation than to positive interactions. The weights representing positive and negative interactions are denoted by γ and δ, respectively. It is noteworthy that $\gamma \ll \delta$, and $\gamma + \delta = 1$.

b) The Current Learning Reputation estimation of client n with Interaction Freshness: In cross-silo FL, the reputation of clients undergoes changes across rounds, rendering the client intermittently unreliable for model training. Similarly, we apply the freshness fading function $\zeta(t)$ to assign weights to clients based on their interaction freshness. Consequently, the current learning reputation of client n up to round t is formulated as Eq. (15).

$$\begin{aligned} \hat{b}_n^{t_c} &= \frac{\sum_{t=t_0}^{t_{c-1}} [\zeta(t) b_n^t]}{\sum_{t=t_0}^{t_{c-1}} \zeta(t)} \\ \hat{d}_n^{t_c} &= \frac{\sum_{t=t_0}^{t_{c-1}} [\zeta(t) d_n^t]}{\sum_{t=t_0}^{t_{c-1}} \zeta(t)} \\ \hat{u}_n^{t_c} &= \frac{\sum_{t=t_0}^{t_{c-1}} [\zeta(t) u_n^t]}{\sum_{t=t_0}^{t_{c-1}} \zeta(t)} \\ \textit{and} \quad \hat{R}_n^{t_c} &= \frac{\sum_{t=t_0}^{t_{c-1}} [\zeta(t) R_n^t]}{\sum_{t=t_0}^{t_{c-1}} \zeta(t)}. \end{aligned} \tag{15}$$

Algorithm 2 Client Selection.

Input: # Positive/negative interactions α_n/β_n, learning quality/reputation set (client n) q_n/R_n, packet transmission success probability p_n, learning quality/reputation set (round t_c) q^{t_c}/R^{t_c}, selected client $\mathcal{N}_s$.
Output: Selected client $\mathcal{N}_s$, normalized contribution N_c.

$L^{all} \leftarrow \emptyset$
for $n \in \mathcal{N}_s$ **do**
  $L_n(w_n; \mathcal{X}_n) \leftarrow \frac{1}{|\mathcal{N}_s|} \sum_{n \in \mathcal{N}_s} l_n(w_n)$
  $L^{all} \leftarrow L^{all} + \{L_n\}$
end for
$loss^{avg} \leftarrow \text{avg}(L^{all})$
$q_{t_c}, R_{t_c} \leftarrow \textbf{QRE}(loss^{avg}, \alpha_n, \beta_n, q_n, R_n, p_n)$
$O_c \leftarrow \frac{q_{t_c}}{\max q_{t_c}} \times \frac{R_{t_c}}{\max R_{t_c}}$
Sort $\mathcal{N}_s$ in descending order of O_c as M_c
for $m \in M_c$ **do**
  Record the index position of m as idx;
  if $O_c[m] < 0$ **then**
    break
  end if
end for
$S_{idx} \leftarrow M_c[idx:]$
$S_c \leftarrow O_c[S_{idx}]$
for $n \in \mathcal{N}_s$ **do**
  if $n \in S_{idx}$ **then**
    $\mathcal{N}_s \leftarrow \mathcal{N}_s - \{n\}$
  end if
end for
Normalize the contribution of selected clients S_c as N_c
return $\mathcal{N}_s, N_c$

Algorithm 1 outlines the quality estimation process and reputation estimation for all clients in round t_c and provides a systematic way to estimate the quality and reputation of clients in a distributed system, based on their past performance. These estimates can be used to make informed decisions about which clients to trust and assign tasks to them.

4.1.3 The Estimation of Contribution Degree. Clients may upload intentionally or unintentionally low-quality model updates, thereby compromising the overall performance of the global model. To ensure the client selection with both high quality and reputation, we devise a measurement method employing a heuristic algorithm, outlined as follows:

$$O_c = \frac{\hat{q}^t}{\max \hat{q}^t} \times \frac{\hat{R}^t}{\max \hat{R}^t} \tag{16}$$

where $\hat{q}^t = \left\{\hat{q}_1^t, \hat{q}_2^t, \cdots, \hat{q}_n^t\right\}$ represents the set of learning quality for clients in round t. The maximum learning quality in round t, denoted as $\max \hat{q}^t$, corresponds to the highest value within this set. Similarly, $\hat{R}^t = \left\{\hat{R}_1^t, \hat{R}_2^t, \cdots, \hat{R}_n^t\right\}$ signifies the set of learning reputation for clients in round t, and $\max \hat{R}^t$ denotes the highest learning reputation value in round t. The contribution degree of clients in round t is denoted as O^c. When O_n^c is less than 0, it

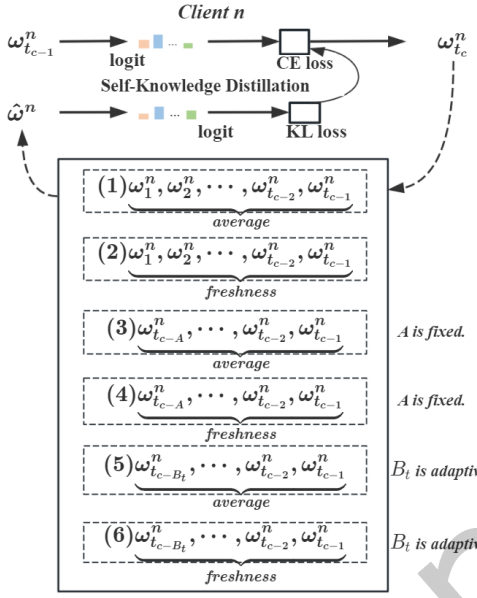


Fig. 3. Adaptive Local Self-Knowledge Distillation with Historical Knowledge.

indicates that client n is deemed unreliable, or the contributed data is not beneficial for enhancing model training performance. In this study, the contribution degree, considering both the quality and reputation of the client, is determined based on the data contributed in the current round and historical records of learning quality and reputation.

Algorithm 2 (Lines 6-8) presents the process of estimating the contribution degree for selected clients in round t_c. The procedure takes into account multiple factors that can impact clients' contribution, such as their data quality, data scale, data heterogeneity, client reputation, and success probability in packet transmission. After estimating the contribution degree for each client, the algorithm excludes those clients whose estimated contribution degree is less than 0 (lines 9-20). This step is crucial for selecting high-quality and high-reputation clients from a long-term perspective, as it eliminates clients who are unlikely to provide significant value to the system in the current round. By focusing on clients with positive estimated contribution degrees, the system can allocate tasks more efficiently, improving overall performance.

4.2 Adaptive Local Self-Knowledge Distillation with Historical Knowledge

Dislike cross-device FL, clients in cross-silo FL may have participated in each round of model training and carried the state in each round, which means the model retains information from previous rounds. To mitigate the negative impact of catastrophic forgetting on performance, we incorporate self-knowledge distillation to guide the current local model using historical knowledge. Specifically, we first generate knowledge from local historical knowledge by combining the local model in each communication round, then transfer the knowledge to the current local model through knowledge distillation. To achieve higher model performance, the key is to explore how to generate a personalized guided model by exploiting local historical knowledge. In this work, we explore six different schemes to identify the most effective scheme for reducing performance degradation due to catastrophic forgetting as illustrated in Figure 3, detailed as follows.

Scheme1: The average of all historical knowledge. In this scheme, the local models of client n are preserved in each round. $W_n = \left\{w_n^0, w_n^1, \cdots, w_n^{t_{c-1}}\right\}$ represents the set of local models in the first t rounds. The local guiding

model for client n in the round t_c is obtained as follows:

$$\hat{w}_n = \frac{\sum_{t=0}^{t_{c-1}} w_n^t}{|W_n|}. \tag{17}$$

Scheme2: All historical knowledge with interaction freshness. Different from Scheme1, the local guiding model for client n in the round t_c in Scheme2 is obtained as follows:

$$\hat{w}_n = \frac{\sum_{t=0}^{t_{c-1}} [\zeta(t) w_n^t]}{\sum_{t=0}^{t_{c-1}} \zeta(t)}. \tag{18}$$

Scheme3: The average of buffer-fixed historical knowledge. In this scheme, the local models of client n are preserved from the last A rounds, where A is a specified fixed number. $F_n = \left\{w_n^{t_{c-A}}, \cdots, w_n^{t_{c-1}}\right\}$ represents the set of local models in the last A rounds. The local guiding model for client n in the round t_c is obtained as follows:

$$\hat{w}_n = \frac{\sum_{t=t_{c-A}}^{t_{c-1}} w_n^t}{|F_n|}. \tag{19}$$

Scheme4: Buffer-fixed historical knowledge with interaction freshness. In contrast to Scheme3, the local guiding model for client n in the round t_c of Scheme4 is in Eq. (20).

$$\hat{w}_n = \frac{\sum_{t=t_{c-A}}^{t_{c-1}} [\zeta(t) w_n^t]}{\sum_{t=t_{c-A}}^{t_{c-1}} \zeta(t)}. \tag{20}$$

Scheme5: The average of buffer-adaptive historical knowledge. In this scheme, the buffer-adaptive scheme employs a distinct historical level B^t for each round t, where B^t is a dynamic and responsive parameter that adjusts with the communication rounds. The following equation succinctly defines the buffer-adaptive historical level B^t:

$$B^t = \begin{cases} 0 & t = 0 \\ \lceil t/b \rceil & t > 0 \end{cases} \tag{21}$$

where b serves as the buffer adaptive parameter, influencing alterations in the buffer size throughout various communication rounds. Further details on this aspect will be provided in Section 5.4. $B_n = \left\{w_n^{t_{c-B^t}}, \cdots, w_n^{t_{c-1}}\right\}$ denotes the set of local models in the last B^t rounds. The local guiding model for client n in round t_c is obtained as follows:

$$\hat{w}_n = \begin{cases} \emptyset & t = 0 \\ \dfrac{\sum_{t=t_{c-B^t}}^{t_{c-1}} w_n^t}{|B_n|} & t > 0 \end{cases} \tag{22}$$

Scheme6: Buffer-adaptive historical knowledge with interaction freshness. Different from Scheme5, the local guiding model for client n in the round t_c of Scheme6 is in Eq. (23):

$$\hat{w}_n = \begin{cases} \emptyset & t = 0 \\ \dfrac{\sum_{t=t_{c-B^t}}^{t_{c-1}} [\zeta(t) w_n^t]}{\sum_{t=t_{c-B^t}}^{t_{c-1}} \zeta(t)} & t > 0 \end{cases} \tag{23}$$

In our evaluation of these schemes, Scheme6 outperformed the other schemes in reducing the impact of catastrophic forgetting. This was demonstrated in the results presented in Section 5.5, which showed that Scheme6 was the most effective scheme at mitigating performance degradation due to catastrophic forgetting.

Then, we utilize self-knowledge distillation from the local guiding model $\hat{w}_n$ to regularize current local training $w_n^{t_c}$. Consequently, the loss function $\Psi_n(w_n^{t_c})$ of the client n in the round t_c is a combination of training loss $L_n(w_n^{t_c})$ and distillation loss as follows:

$$\Psi_n(w_n^{t_c}) = \underbrace{L_n(w_n^{t_c})}_{CE\ Loss} + \underbrace{\lambda \mathcal{L}_{KL}(p(\hat{w}_n)||p(w_n^{t_c}))}_{KD\ Loss}. \tag{24}$$

where λ is a parameter that controls the relative weight of the two terms in the loss function and higher values of λ mean placing more emphasis on knowledge distillation. Here, $L_n(\cdot)$ denotes the standard cross-entropy loss function while $\mathcal{L}KL(\cdot)$ signifies the Kullback-Leibler (KL) Divergence between the guiding personalized prediction $p(\hat{w}_n)$ and the current local prediction $p(w_n t_c)$. $p(\cdot)$ represents a softmax function that calculates the soft prediction.

In the ASDQR framework, each client n updates its local weights $w_n^{t_c}$ by running stochastic gradient descent (SGD) using its local objective $\Psi_n(w_n^{t_c})$ instead of $L_n(w_n^{t_c})$. More specifically, the local weights $w_n^{t_c}$ are updated using the following SGD update rule:

$$w_n^{t_c} = w_n^{t_c} - \eta \nabla \Psi_n(w_n^{t_c}, \hat{w}_n^{t_c}) \tag{25}$$

where η denotes the learning rate, and $\nabla \Psi_n(w_n^{t_c})$ represents the gradient of the local objective concerning the local weights.

To address resource limitations, we developed the ClientBufferSelection and UpdateBuffer algorithms, as detailed in Appendix D. These algorithms enable users to select an appropriate buffering strategy based on their available resources. For clients with substantial resources, we recommend using larger adaptive buffers (e.g., ABuffer15) to improve model performance and personalization. Conversely, for clients with limited resources, a small fixed buffer (e.g., Buffer1) is more suitable, as it reduces computational and storage overhead.

4.3 Inference Effective Aggregation Scheme

From the analysis above, we observe that FedAvg focusing only on data quantity can lead to several challenges, especially when dealing with data heterogeneity. To solve the challenges, we incorporate data quality, data scale, data heterogeneity, distribution difference, and client reputation when performing global model aggregation. Specifically, We utilize the contribution degree obtained by estimated quality and reputation as aggregation weights to give higher weights to important, high-quality, and reliable model updates.

$$w^{t_c+1} \leftarrow \frac{\sum_{n \in \mathcal{N}_s} [N_n^c w_n^{t_c,K}]}{\sum_{n \in \mathcal{N}_s} N_n^c} \tag{26}$$

where $\mathcal{N}_s$ means the selected clients, N_n^c denotes the Normalized contribution degree, $w_n^{t_c,K}$ presents the local model updates distilled from historical knowledge, details in Section 4.2 Scheme6.

The procedure for local client update of the ASDQR framework is presented in Algorithm 3 (Lines 20-30). Each client n updates its local weights $w_n^{t_c}$ by using its local objective $\Psi_n(w_n^{t_c})$ instead of $L_n(w_n^{t_c})$. The server then aggregates the local updates and assigns weights according to their contribution degree, as detailed in Algorithm 3 (Lines 5-19). ClientBufferSelection in Line 3 offers resource-aware dynamic buffer selection, ensuring adaptation to diverse client resource conditions, while UpdateBuffer in Line 16 improves model training generalization and stability through efficient historical update management. Together, they empower the algorithm with robust adaptability, resilience, and communication efficiency in heterogeneous and non-IID environments. Furthermore, these mechanisms enhance communication efficiency in Appendix E by minimizing redundant transmissions and adaptively managing data, while optimizing Computational Complexity in Appendix F through dynamic adjustments of buffer size and computational workload, ensuring scalability across varied client resources.

Algorithm 3 ASDQR with Buffer Selection.

Input: The set of clients $\mathcal{N}$, step size η, the success probability of packet transmission p_n.
Output: Global model w^{t_c+1}.

for each client $n \in \mathcal{N}$ **do**
  $buffer_n \leftarrow \emptyset, q_n \leftarrow \emptyset, R_n \leftarrow \emptyset, \alpha_n \leftarrow 0, \beta_n \leftarrow 0$
  $buffer\_type, fixed\_size, b \leftarrow$ ClientBufferSelection(resources$_n$) ▷ Appendix D
end for
for $t_c = 0$ to $T-1$ **do**
  if $t_c = 0$ **then**
    $\mathcal{N}_s \leftarrow \mathcal{N}$
  else
    $\mathcal{N}_s, N_c \leftarrow$ **Client Selection**$(\alpha_n, \beta_n, q_n, R_n, p_n)$
  end if
  Cloud server sends w^{t_c} to selected clients $\mathcal{N}_s$
  $w_n^{t_c,0} \leftarrow w^{t_c}$
  for each client $n \in \mathcal{N}_s$ in parallel **do**
    $w_n^{t_c,K} \leftarrow ClientUpdate(t_c, buffer_n, w_n^{t_c,0})$
    $buffer_n \leftarrow buffer_n + \{w_n^{t_c,K}\}$
    UpdateBuffer$(buffer_n, w_n^{t_c,K}, buffer_type, t_c, fixed_size, b)$ ▷ Appendix D
  end for
  Aggregates local updates: $w^{t_c+1} \leftarrow \frac{\sum_{n\in\mathcal{N}_s} N_n^c w_n^{t_c,K}}{\sum_{n\in\mathcal{N}_s} N_n^c}$
end for
function ClientUpdate$(t_c, buffer_n, w_n^{t_c,0})$
  if $t_c = 0$ **then**
    $\hat{w}_n^{t_c} = 0$
  else
    $$\hat{w}_n^{t_c} = \frac{\sum_{t=t_{c-B^t}}^{t_{c-1}} \zeta(t) \times buffer_n[t]}{\sum_{t=t_{c-B^t}}^{t_{c-1}} \zeta(t)}$$
  end if
  for $k = 0$ to $K-1$ **do**
    Computes gradient: $w_n^{t_c,k} \leftarrow \nabla L_n(w_n^{t_c,k}, x_n^{t_c,k})$
    $w_n^{t_c,k+1} \leftarrow w_n^{t_c,k} - \eta \nabla \Psi_n(w_n^{t_c,k}, \hat{w}_n^{t_c})$
  end for
end function

Through this process, the server allocates weight to each client's model update, considering its contribution degree, which incorporates both the learning quality and reliability of the model update. This approach ensures that updates with higher contribution degrees receive increased weight in the aggregation process, thereby improving the overall performance of the global model. By utilizing the estimation of data quality and client reliability, the ASDQR framework can also effectively handle non-IID data challenges. In addition, we present a theoretical convergence analysis of Algorithm 3 for strongly convex problems, considering practical assumptions such as non-IID, partial device participation, and local updating. The proof is shown in Appendix B.

Table 2. Inference Accuracy Overview Under Non-IID.

Dataset	Method	Scale	FedAvg	MOON	FedProx	PFedSD	FedMD	ASDQR
MNIST	MLP	10	93.54 ± 0.06	93.82 ± 0.33	95.94 ± 0.12	95.73 ± 0.07	95.98 ± 0.09	**96.04 ± 0.09**
		50	90.28 ± 0.13	90.32 ± 0.12	93.05 ± 0.16	93.00 ± 0.12	93.12 ± 0.11	**93.43 ± 0.04**
		100	88.51 ± 0.11	88.54 ± 0.07	91.32 ± 0.12	91.31 ± 0.12	91.49 ± 0.11	**91.72 ± 0.07**
	CNN	10	96.31 ± 0.16	96.43 ± 0.20	98.33 ± 0.09	98.34 ± 0.13	98.47 ± 0.07	**98.53 ± 0.04**
		50	92.17 ± 0.38	96.21 ± 0.17	96.30 ± 0.07	92.30 ± 0.13	96.53 ± 0.09	**96.68 ± 0.03**
		100	85.40 ± 0.30	89.75 ± 0.27	93.44 ± 0.16	93.31 ± 0.06	94.21 ± 0.12	**94.59 ± 0.04**
	ResNet18	10	99.19 ± 0.06	99.35 ± 0.06	99.21 ± 0.03	99.40 ± 0.01	99.41 ± 0.04	**99.44 ± 0.03**
		50	99.08 ± 0.03	99.33 ± 0.06	99.11 ± 0.15	99.38 ± 0.07	99.40 ± 0.05	**99.42 ± 0.02**
		100	99.01 ± 0.09	99.23 ± 0.11	99.06 ± 0.12	99.30 ± 0.11	99.32 ± 0.10	**99.36 ± 0.09**
CIFAR-10	MLP	10	43.70 ± 0.63	44.68 ± 0.09	47.54 ± 0.42	47.15 ± 0.70	48.11 ± 0.13	**48.27 ± 0.24**
		50	39.09 ± 0.16	42.30 ± 0.25	43.40 ± 0.46	43.18 ± 0.16	43.47 ± 0.19	**43.56 ± 0.15**
		100	35.95 ± 0.24	40.50 ± 0.23	40.54 ± 0.22	40.45 ± 0.34	41.09 ± 0.17	**41.27 ± 0.05**
	CNN	10	56.56 ± 1.14	59.83 ± 0.89	67.62 ± 1.30	68.15 ± 0.15	68.42 ± 0.17	**68.61 ± 0.11**
		50	42.38 ± 0.34	54.13 ± 0.32	55.22 ± 0.10	55.27 ± 0.64	55.68 ± 0.23	**55.96 ± 0.12**
		100	30.89 ± 0.29	42.15 ± 1.40	43.95 ± 0.54	44.23 ± 0.66	46.93 ± 0.58	**47.45 ± 0.08**
	ResNet18	10	63.71 ± 0.96	66.43 ± 0.61	66.24 ± 0.94	71.29 ± 0.85	73.29 ± 0.51	**75.63 ± 0.30**
		50	50.31 ± 0.26	59.86 ± 0.66	52.56 ± 0.72	62.99 ± 0.90	64.28 ± 0.87	**66.33 ± 0.71**
		100	42.70 ± 0.30	55.28 ± 0.56	44.59 ± 0.31	56.37 ± 0.49	57.39 ± 0.49	**58.50 ± 0.60**
HAR	MLP	10	82.31 ± 1.55	84.74 ± 0.66	85.71 ± 0.51	86.08 ± 0.40	86.91 ± 0.53	**87.71 ± 0.29**
		50	61.93 ± 0.60	75.84 ± 0.64	77.00 ± 0.85	79.46 ± 1.16	81.17 ± 0.39	**82.64 ± 0.04**
		100	55.61 ± 0.19	60.40 ± 1.60	61.16 ± 1.29	60.63 ± 1.43	62.84 ± 0.61	**64.52 ± 0.03**
	CNN	10	80.19 ± 2.66	90.05 ± 1.71	89.24 ± 0.48	90.75 ± 0.11	90.96 ± 0.08	**91.15 ± 0.05**
		50	63.28 ± 0.53	75.94 ± 0.22	78.25 ± 0.28	79.61 ± 0.36	81.74 ± 0.19	**83.13 ± 0.12**
		100	59.51 ± 1.43	65.80 ± 0.68	69.50 ± 0.95	70.03 ± 0.56	71.24 ± 0.58	**72.06 ± 0.14**
	ResNet18	10	90.56 ± 0.39	93.63 ± 1.71	92.31 ± 0.36	93.90 ± 1.60	93.84 ± 0.87	**94.51 ± 0.63**
		50	89.59 ± 2.22	91.91 ± 0.43	91.21 ± 0.24	92.01 ± 1.03	92.06 ± 0.69	**92.56 ± 0.78**
		100	83.95 ± 0.45	89.51 ± 1.37	88.33 ± 0.33	90.53 ± 0.63	91.21 ± 0.74	**91.93 ± 0.46**
WISDM	MLP	10	74.53 ± 0.14	79.74 ± 0.92	80.38 ± 0.27	80.18 ± 0.35	81.52 ± 0.29	**82.71 ± 0.40**
		50	67.75 ± 0.39	77.62 ± 2.78	76.11 ± 0.53	76.90 ± 0.23	78.16 ± 0.72	**78.94 ± 0.38**
		100	62.15 ± 0.47	70.26 ± 0.13	70.97 ± 0.61	71.42 ± 0.47	72.06 ± 0.39	**73.20 ± 0.45**
	CNN	10	79.62 ± 0.31	85.13 ± 0.54	84.83 ± 0.10	88.02 ± 1.02	89.69 ± 0.32	**90.84 ± 0.09**
		50	68.05 ± 1.26	80.21 ± 0.37	79.03 ± 0.49	81.08 ± 0.67	82.95 ± 0.41	**84.26 ± 0.14**
		100	61.78 ± 0.89	63.19 ± 0.34	62.41 ± 0.01	65.78 ± 0.29	67.63 ± 0.29	**68.99 ± 0.15**
	ResNet18	10	83.79 ± 1.27	87.44 ± 0.50	88.04 ± 0.86	92.12 ± 0.99	93.37 ± 0.67	**95.64 ± 0.32**
		50	81.57 ± 0.07	87.09 ± 1.06	84.83 ± 1.00	89.86 ± 0.39	89.72 ± 0.14	**90.23 ± 0.03**
		100	79.59 ± 3.23	86.39 ± 0.34	83.44 ± 0.82	87.23 ± 1.53	89.62 ± 0.49	**90.18 ± 0.05**

4.4 Secure Analysis

In this subsection, we present a security analysis of the ASDQR framework, detailing potential security threats and the balance between privacy preservation and system efficiency. Security Implications of ASDQR are as follows:

(1) Privacy Preservation: ASDQR utilizes the FL paradigm to ensure data privacy by only uploading model updates, not raw data. The framework's adaptive self-knowledge distillation further anonymizes these updates by incorporating personalized historical knowledge, thereby diminishing the risk of reverse engineering attacks.
(2) Robustness Against Poisoning Attacks: ASDQR's client selection, based on quality scores and reputation, mitigates the risks of model poisoning. This selective participation reduces the likelihood of malicious updates, enhancing the model's resilience against data poisoning and backdoor attacks.

(3) Data Integrity and Model Security: The framework's adaptive local self-knowledge distillation constrains the impact of any single client's update on the global model, protecting against compromised or malicious clients. The inference-effective aggregation, which assigns weights based on client contributions, prevents the disproportionate influence of anomalous updates.

While ASDQR demonstrates significant improvements in communication efficiency and inference accuracy across diverse datasets and FL scenarios, several limitations require further exploration. One key challenge is addressing extreme client heterogeneity. When clients exhibit substantial disparities in computational resources, network conditions, or data distributions, the current adaptive mechanisms may struggle to maintain fairness and robustness. Moreover, under stringent communication constraints, even optimized frequent model updates can become a bottleneck. Additionally, real-time deployments may face challenges due to latency introduced by adaptive client selection and aggregation processes. The framework's implementation complexity also necessitates careful tuning of components, such as the reputation system, adaptive knowledge distillation, and model aggregation. Future research could explore advanced client selection strategies, including data-driven approach, stratified sampling, clustering, or fairness-aware algorithms, to enhance client diversity and improve global model generalization while ensuring fairness. Augmenting the knowledge distillation framework with hierarchical or graph-based methods may better accommodate highly non-IID data distributions. Finally, adopting optimized communication strategies, such as gradient compression, quantization, or asynchronous protocols, could significantly reduce communication overhead, improving the framework's scalability in resource-constrained environments.

5 Evaluations

In this section, we evaluate the performance of ASDQR to train different models on a variety of datasets and discuss the experimental results. Here, we set frac =60% to reduce the performance degradation of other baselines due to random selection. We assume three different FL scenarios: 1) $N = 10$ clients with frac = 60% sampleing ratio; 2) $N = 50$ clients with frac = 60% sampleing ratio; 3) $N = 100$ clients with frac = 60% sampleing ratio. We run $T = 100$ communication rounds for every scenario. During each training round, all clients adopt a fixed mini-batch size of 32 and perform local training iterations with a fixed value of K=5. Moreover, we set the temperature for soft prediction to 5. Plus, we set the noisy level of data with {0, 0.5, 1} by $label(d) = (label(d) + 1)\%10$, where d is the true label. To evaluate the performance of each method, we record the average inference accuracy and loss across all participating clients.

5.1 Experimental Setup

Baseline. For a fair comparison, we will evaluate ASDQR against three baseline methods:

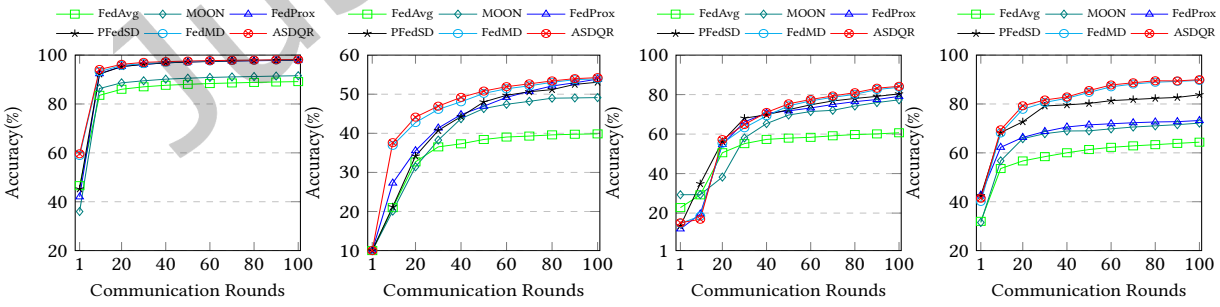
- **FedAvg.** FedAvg [32] was introduced as a pioneering work in FL, involving the random selection of participating clients and the aggregation of their local model updates to train a global model.
- **MOON.** MOON [25] is a model-contrastive FL framework that enhances the performance of classification tasks by leveraging the similarity between model representations to correct local training across clients.
- **FedProx.** FedProx [27] addresses the challenge of non-IID data in FL by introducing a proximal term to the local objective function. This addition ensures that client updates do not deviate excessively from the global model.
- **PFedSD.** PFedSD [18] leverages the value of historical personalized models in FL with the use of self-distillation, overcoming the problem of forgetting and achieving a more favorable balance between personalization and generalization.
- **FedMD.** FedMD [24] employs knowledge distillation and transfer learning to facilitate collaboration among clients with independently designed local models. By utilizing a public dataset as a communication medium, FedMD enhances the accuracy of individual models while maintaining data privacy and protecting model architecture details.

Models & Datasets. To demonstrate the broad applicability of ASDQR, we selected two models and datasets for our experiments. Specifically, we employed ASDQR to train Multilayer Perceptron (MLP), Convolutional Neural Network (CNN), and Residual Network 18 (ResNet18) on different datasets, including MNIST [21], CIFAR-10 [20], HAR [34] and WISDM [42]. For the CIFAR-10 and MNIST datasets, MLP models are configured with input dimensions determined by the product of the image dimensions—3072 for CIFAR-10 (3x32x32 images) and 784 for MNIST (1x28x28 images). Both models utilize a hidden layer of 64 neurons and feature a variable number of output classes based on dataset-specific requirements. In contrast, the MLP model designed for the HAR and WISDM dataset is configured with an input dimension of 1152, also uses a 64-neuron hidden layer, and outputs to 6 classes. In the realm of CNN, the architecture for MNIST includes two convolutional and two fully connected layers, incorporating dropout and pooling to optimize image processing. The CNN designed for CIFAR features, being more complex, includes three convolutional layers with ReLU and max-pooling, followed by two fully connected layers for final classification. Meanwhile, the CNN for HAR and WISDM consists of three convolutional layers with ReLU and max-pooling, followed by a flattened output passed through a dense layer with dropout for classification. In ResNet18, the initial convolutional layer is adapted to the dataset using a pre-trained ResNet18 model, with the original fully connected layer replaced by a linear layer of a specified encoding length, followed by a linear classifier to categorize the encoded features.

5.2 Inference Accuracy

We examine the performance of various FL methods in handling non-IID data distributions across diverse datasets and model architectures while maintaining a constant total data volume, as depicted in Table 2. The ASDQR inference accuracy is compared with several baseline approaches in a non-IID setting where the data partition ratio is [6:5:4:3:2]. According to the results presented in Table 2, ASDQR consistently outperforms FedAvg, MOON, FedProx, PFedSD, and FedMD across all datasets, methods, and client scales under non-IID settings. Notably, the accuracy of all methods decreases as the number of participating clients increases, but ASDQR demonstrates its effectiveness in mitigating the adverse effects of non-IID data, resulting in superior inference accuracy.

Figure 4 compares the performance of the proposed ASDQR approach with baseline methods in terms of inference accuracy over varying communication rounds, considering 100 participating clients using a CNN model. The results demonstrate that ASDQR consistently outperforms baseline methods, including FedAvg, MOON, FedProx, PFedSD, and FedMD, across diverse datasets (MNIST, CIFAR, HAR, and WISDM). For instance, on the MNIST dataset at 100 communication rounds, the inference accuracies for FedAvg, MOON, FedProx, PFedSD, FedMD, and ASDQR are 89.16%, 91.57%, 97.87%, 97.96%, 98.08%, and 98.16%, respectively. Notably, ASDQR achieves higher accuracy and exhibits faster convergence compared to the other methods. The accuracy improvement



(a) MNIST (b) CIFAR (c) HAR (d) WISDM

Fig. 4. The average accuracy of the learning model with different rounds when the number of clients is 100 under different noisy levels.

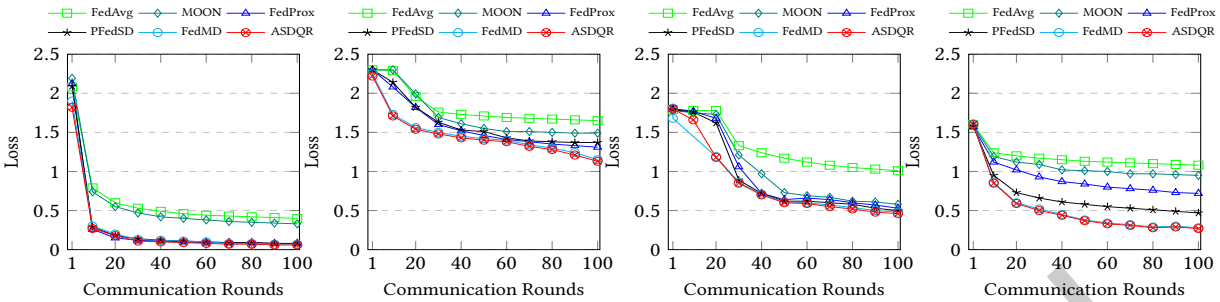


Fig. 5. The average loss of the learning model with different rounds when the number of clients is 100 under different noisy levels.

trend for ASDQR on MNIST is particularly pronounced, starting from 59.62% at round 1 and reaching 98.16% at round 100. By contrast, FedAvg improves from 46.57% to 89.16%, MOON from 35.99% to 91.57%, FedProx from 41.99% to 97.87%, PFedSD from 45.23% to 97.96%, and FedMD from 58.92% to 98.08%. The performance on the CIFAR dataset shows a similar trend, albeit with a lower accuracy range due to the dataset's higher complexity. ASDQR achieves the highest accuracy at 54.24% after 100 communication rounds, compared to 39.85% for FedAvg, 49.13% for MOON, 53.86% for FedProx, 53.06% for PFedSD, and 53.93% for FedMD. For HAR, ASDQR demonstrates significant accuracy growth, starting from 15.10% in the initial round and reaching 84.19% after 100 rounds. It outperforms the baseline methods, such as FedAvg (60.67%), MOON (77.35%), FedProx (79.11%), PFedSD (80.36%), and FedMD (83.71%). On the WISDM dataset, ASDQR achieves a final accuracy of 89.96%, compared to 64.37% for FedAvg, 72.17% for MOON, 73.23% for FedProx, 83.77% for PFedSD, and 89.63% for FedMD.

These results clearly illustrate ASDQR's ability to achieve higher accuracies across diverse datasets and scenarios consistently. Furthermore, the model's adaptability across various architectures such as MLP, CNN, and ResNet18 highlights its versatility in FL tasks.

5.3 Convergence Speed

To validate the effectiveness of our theoretical analysis, we compared the convergence rates of four different FL methods: FedAvg, FedProx, PFedSD, FedMD, and ASDQR.

Figure 5 presents the average loss trends of various methods (FedAvg, MOON, FedProx, PFedSD, FedMD, and ASDQR) across different datasets (MNIST, CIFAR, HAR, and WISDM) with 100 clients over varying communication rounds. The results indicate that ASDQR consistently achieves faster convergence and lower final loss compared to all baseline methods. On MNIST, ASDQR reduces the loss from 1.82 in the first round to 0.06 after 100 rounds, outperforming FedAvg (0.40), MOON (0.33), FedProx (0.08), PFedSD (0.08), and FedMD (0.07). Although FedMD demonstrates competitive results, ASDQR converges more quickly and achieves a slightly lower final loss. On CIFAR, a more challenging dataset, ASDQR achieves a final loss of 1.13, which is significantly lower than that of FedAvg (1.65), MOON (1.49), FedProx (1.31), PFedSD (1.37), and marginally better than FedMD (1.15). Its superior convergence speed highlights its efficiency under complex conditions. For HAR, ASDQR achieves the lowest final loss (0.46), surpassing FedAvg (1.01), MOON (0.58), FedProx (0.53), PFedSD (0.50), and FedMD (0.48). ASDQR maintains a consistently faster convergence rate and greater stability throughout the training process. On WISDM, ASDQR reduces the loss from 1.60 initially to 0.27 at 100 rounds, outperforming FedAvg (1.08), MOON (0.95), FedProx (0.72), PFedSD (0.47), and FedMD (0.28). Its rapid and stable convergence further underscores its superiority in this scenario.

ASDQR demonstrates consistent advantages in loss reduction and convergence speed across all datasets. These results underscore its robustness and efficiency in FL scenarios with noisy and non-IID data distributions,

establishing it as an effective solution for optimizing model training in distributed environments. Additionally, we have provided proof of convergence for ASDQR in Appendix B.

5.4 Inference Accuracy vs.Communication Cost

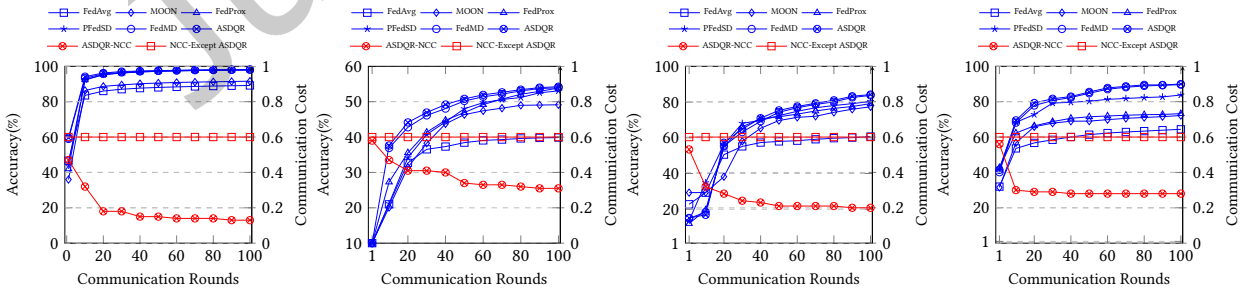
Figure 6 provides a comprehensive comparison between ASDQR and baseline methods in terms of normalized communication cost and inference accuracy. Results are derived from a CNN model with 100 clients. Each blue curve in Figure 6 corresponds to the accuracy of a distinct FL method, namely FedAvg, MOON, FedProx, PFedSD, FedMD, and ASDQR. The ASDQR curve's upward trajectory signifies its superior accuracy performance. In contrast, the red curves illustrate the *normalized communication cost* (NCC), with ASDQR-NCC and NCC-Except ASDQR showing distinct patterns. Notably, the NCC-Except ASDQR curve reflects a fixed normalized communication cost of 0.6 for the baseline methods, including FedAvg, FedProx, PFedSD, and FedMD, which employ a random selection strategy of 60 clients from the pool of 100 in each round. This method often results in inefficiencies, as the static selection fails to adapt to the dynamic nature of the learning process. Conversely, the descending trajectory of the ASDQR-NCC curve underscores its communication efficiency, indicating that ASDQR optimally utilizes communication resources and adapts to the variability in client contributions more effectively than the NCC-Except ASDQR.

ASDQR consistently outperforms baseline methods across all datasets. Notably, ASDQR achieves the highest accuracy of 98.16% on MNIST, 54.24% on CIFAR, 84.19% on HAR, and 89.96% on WISDM. These results demonstrate its robustness and adaptability, particularly in complex and heterogeneous data scenarios. While baseline methods (NCC-Except ASDQR) maintain a static NCC of 0.6 due to random client selection, ASDQR dynamically reduces communication costs over time. The ASDQR-NCC curve exhibits a descending trend across datasets, converging to 0.13-0.2 by the 100th round, reflecting its adaptive strategy to prioritize high-contribution clients and minimize redundant communication. ASDQR's dynamic communication mechanism significantly reduces resource consumption without compromising model performance. This adaptability is critical in FL scenarios, particularly in resource-constrained environments where efficient bandwidth utilization is essential.

ASDQR achieves a compelling balance between inference accuracy and communication efficiency, outperforming traditional static approaches in FL. Its adaptive strategy ensures scalability and robustness, making it an effective solution for dynamic and distributed learning environments.

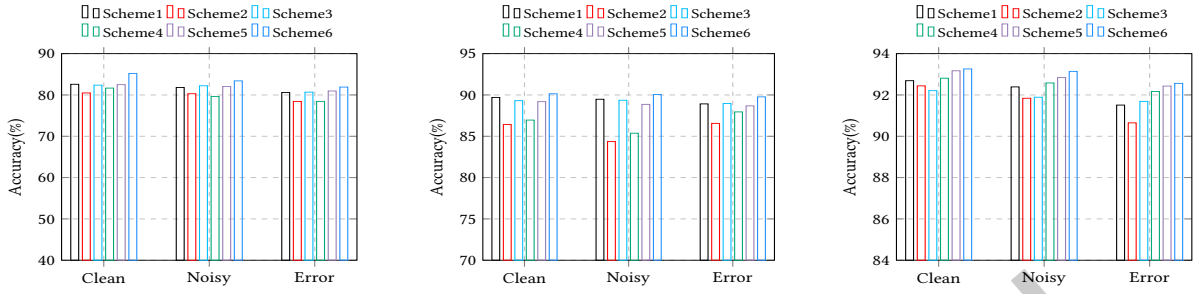
5.5 Evaluation of Six Schemes for Generating Guiding Model

We investigate the impact of six different schemes for generating guiding models on the average accuracy of neural network architectures, including MLP, CNN, and ResNet18, under varying noise levels, with 100 clients. The results are depicted in Figure 7.



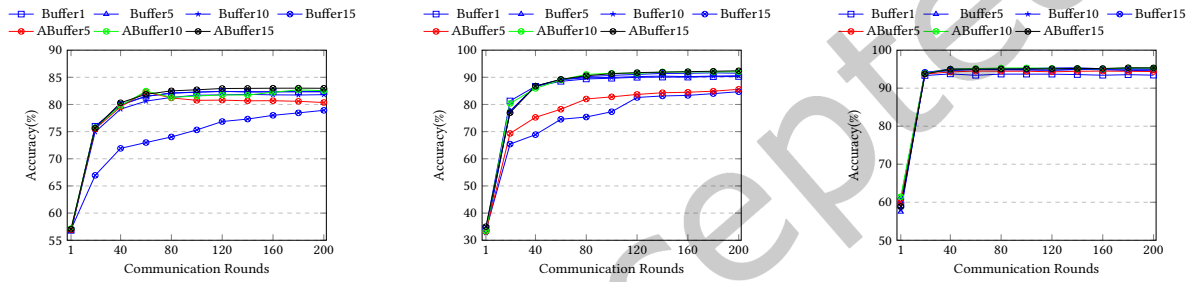
(a) MNIST (b) CIFAR (c) HAR (d) WISDM

Fig. 6. Comparison Between ASDQR and Baselines in Communication Cost and Inference Accuracy.



(a) MLP (b) CNN (c) ResNet18

Fig. 7. The average accuracy of the learning model with different rounds when the number of clients is 100 under different noisy levels.



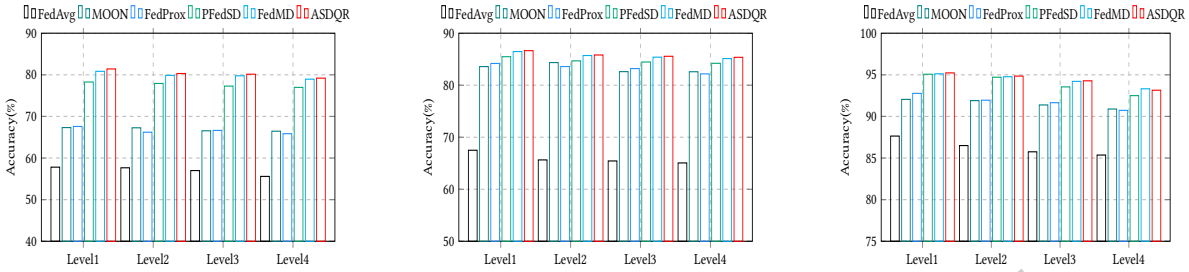
(a) MLP (b) CNN (c) ResNet18

Fig. 8. Buffer-adaptive self-knowledge distillation.

Both Scheme 6 and Scheme 5 consistently demonstrate superior performance and greater stability across all architectures compared to the other schemes. This consistent trend highlights the effectiveness of buffer-adaptive historical knowledge, which leverages a dynamic parameter B^t that adjusts based on the communication rounds, allowing the models to respond more effectively to varying noise levels in the data. Furthermore, Scheme 6 consistently outperforms Scheme 5, underscoring the impact of the integrated interaction freshness mechanism. This mechanism, represented by the function $\zeta(t)$, enhances the model's ability to prioritize recent and relevant historical information, thereby improving its decision-making process. The design of Scheme 6 enables a more refined understanding of the data, especially in noisy label conditions that could otherwise distort learning. Across all neural network architectures tested, Scheme 6 consistently delivers higher accuracy, suggesting that both buffer adaptation and interaction freshness are versatile and effective enhancements.

5.6 Adaptive Buffer

Figure 8 presents the results of buffer-adaptive self-knowledge distillation experiments conducted on MLP, CNN, and ResNet18. Specifically, Buffer1, Buffer5, Buffer10, and Buffer15 correspond to fixed buffer sizes of 1, 5, 10, and 15, respectively. In contrast, ABuffer5, ABuffer10, and ABuffer15 represent adaptive buffer configurations, where the parameter b in Eq. (21) is set to 5, 10, and 15, respectively. As depicted in Figure 8, ABuffer15 outperforms all other schemes, including fixed buffer sizes and smaller adaptive buffers, underscoring its efficacy. Generally, ABuffer10 exhibits superior performance compared to both ABuffer5 and fixed buffer schemes (Buffer1, Buffer5, Buffer10, Buffer15). Remarkably, fixed buffer schemes demonstrate better performance than adaptive buffer scheme ABuffer5. The performance difference between ABuffer15, ABuffer5, and ABuffer10 can be attributed to the dynamics of the buffer-adaptive historical level B^t, which is determined by the buffer adaptive parameter



(a) MLP (b) CNN (c) ResNet18

Fig. 9. The Impact of Unbalanced Level on Performance.

b. A smaller b induces a more rapid adaptation of the buffer size to recent changes, enhancing the model's responsiveness to variations in the data distribution or model updates. However, this expedited adaptation may come at the cost of less stable historical knowledge, potentially capturing short-term fluctuations. Conversely, a larger b slows down the adaptation process, furnishing a more stable historical knowledge base over time. While this stability aids in filtering out noise or short-term variations, it might render the model less responsive to recent changes. Therefore, the selection of b in the buffer-adaptive mechanism is pivotal, representing a delicate trade-off between responsiveness and stability.

For MLP, as shown in Figure 8 (a), ABuffer10 and ABuffer15 exhibit slightly better performance than the fixed buffers after 200 communication rounds, achieving accuracy rates of 82.54% and 82.98%, respectively. Notably, excessively large buffers can increase storage and computational demands, potentially affecting training speed and efficiency. For example, in the first round of communication, the accuracy rates for Buffer1, Buffer5, Buffer10, Buffer15, ABuffer5, ABuffer10, and ABuffer15 were 56.82%, 56.68%, 56.77%, 57.00%, 56.91%, 57.14%, and 57.05%, respectively. By the 200th round, these rates had improved to 82.83%, 82.38%, 81.80%, 78.92%, 80.36%, 81.54%, and 82.98%, respectively. Based on this analysis, we selected $b = 15$ for our study. In the early stages of training, fixed buffer strategies may outperform adaptive buffers; however, as the number of communication rounds increases, adaptive buffers generally demonstrate superior performance. This trend is also observed in CNN and ResNet18, as shown in Figure 8 (b) and Figure 8 (c). For more detailed information, please refer to Table 4, Table 5, and Table 6 in Appendix C.

5.7 The Impact of Data Heterogeneity

Data heterogeneity is a prevalent scenario in FL, as each client collects data from its unique local environment, leading to significant variability. Simulating this data heterogeneity is essential for assessing the real-world applicability and effectiveness of FL algorithms, as it enables the evaluation of their robustness under realistic conditions. In this work, we simulate data heterogeneity through data partitioning, allocating time-varying subsets according to the number of clients N and the ratio rt. We define four unbalanced levels—[16:12:8:4:1] (unbalanced level 4), [12:9:6:3:1] (unbalanced level 3), [8:6:4:2:1] (unbalanced level 2), and [1:1:1:1:1] (unbalanced level 1)—for each round t. The unbalanced level is determined by the ratio of the amounts of data, with higher values indicating a greater degree of imbalance. We also include the balanced case (unbalanced level 1) for comparative purposes. In Figure 9, we illustrate the average accuracy of the learning model across various communication rounds, utilizing the HAR dataset with 100 clients under different unbalanced levels, considering three model architectures: MLP, CNN, and ResNet18. The results are presented for five FL methods: FedAvg, MOON, FedProx, PFedSD, and our proposed method, ASDQR. Notably, for the MLP model, ASDQR consistently demonstrated superior accuracy across all levels of data imbalance, achieving accuracies of 81.41%, 80.31%, 80.14%, and 79.20% for unbalanced levels 1 through 4, respectively. This trend is similarly observed with the CNN model,

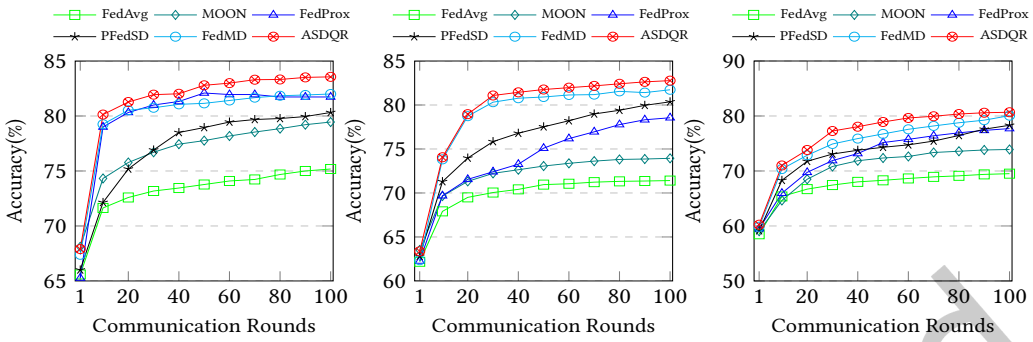


Fig. 10. The average accuracy across varying number of clients (50, 100, and 200 Clients) Over 100 Communication Rounds

where ASDQR recorded commendable accuracy levels of 86.84%, 85.81%, 85.56%, and 85.36% across the same levels. In the case of ResNet18, ASDQR again outperformed other methods, with accuracies of 95.23%, 94.85%, 94.28%, and 93.15% for levels 1 through 4.

These results clearly illustrate that ASDQR consistently outperforms FedAvg, MOON, FedProx, and PFedSD across all three models and varying levels of data imbalance. Such consistent performance highlights the robustness and reliability of ASDQR in FL, particularly when confronted with varying degrees of data heterogeneity. These findings underscore ASDQR's potential as a leading FL method, emphasizing its significance in practical applications where data imbalance is a critical factor.

5.8 Scalability Testing

To assess ASDQR's scalability, we conducted experiments with varying client pool sizes (50, 100, and 200 clients) over 100 communication rounds using the Wisdm dataset on MLP model. This evaluation aims to analyze the framework's performance in terms of inference accuracy, communication efficiency, and model convergence under increasing client numbers.

As shown in Figure 10, in experiments with 50, 100, and 200 clients, ASDQR achieves accuracy levels of 83.56%, 82.77%, and 80.68%, respectively, after 100 communication rounds, consistently outperforming baseline methods such as FedAvg, FedProx, PFedSD, MOON, and FedMD. This demonstrates its scalability and ability to maintain competitive accuracy even with increasing numbers of clients. Notably, ASDQR reaches near-final accuracy (70–80%) within the first 20 communication rounds, underscoring its rapid convergence and superior communication efficiency. The model stabilizes within 50 rounds for smaller client pools and approximately 70 rounds for larger ones, showcasing its adaptability to varying client sizes and its convergence stability in large-scale FL scenarios. Additionally, its design emphasizes feature alignment and local loss optimization, enhancing its resilience to noisy environments and further solidifying its applicability. These findings establish ASDQR as a scalable, efficient, and robust framework for FL in diverse and challenging settings.

5.9 Ablation Study

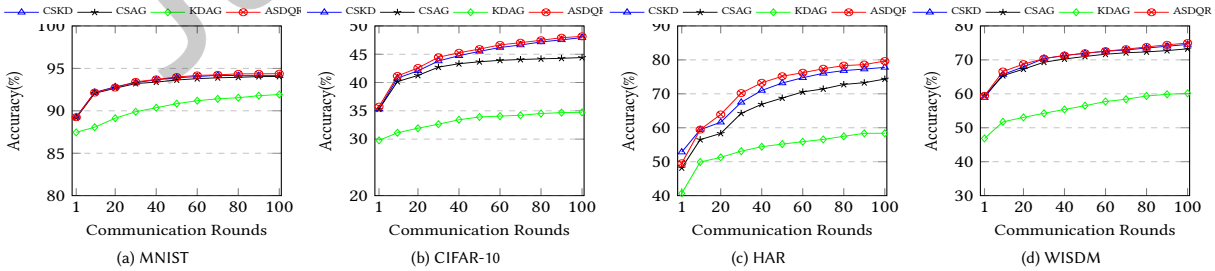
In this section, we analyze the impact of individual design components in ASDQR by systematically removing each component and evaluating its contribution to overall performance. This ablation study provides insights into the effectiveness of key features, including client selection, adaptive self-knowledge distillation, and aggregation optimization.

- **ASDQR.** The full adaptive self-knowledge distillation-based quality- and reputation-aware cross-silo FL framework. It incorporates the proposed client selection mechanism, adaptive self-knowledge distillation using historical knowledge, and aggregation optimization, designed to enhance both FL quality and communication efficiency.
- **CSKD.** A variant of ASDQR without the aggregation optimization component, assessing the impact of aggregation strategies on performance.
- **CSAG.** A variant of ASDQR without the adaptive self-knowledge distillation component, focusing on the contribution of historical knowledge integration.
- **KDAG.** A variant of ASDQR without the client selection component, highlighting the role of quality- and reputation-aware client selection in FL.

Figure 11 compares the performance of the complete ASDQR model with its ablated variants across communication rounds on MNIST, CIFAR-10, HAR, and WISDM datasets using an MLP model. The results demonstrate that ASDQR consistently achieves the highest accuracy across all datasets, highlighting the importance of its components in improving FL quality and communication efficiency. On MNIST, ASDQR achieves 94.37% accuracy, surpassing CSKD by 0.2%, CSAG by 0.33%, and KDAG by 2.45%. For CIFAR-10, a more challenging dataset, ASDQR reaches 48.16% accuracy, exceeding CSKD by 0.22%, CSAG by 3.75%, and KDAG by 13.42%. On HAR, ASDQR achieves 79.65%, outperforming CSKD by 1.86%, CSAG by 5.3%, and KDAG by 21.26%. Lastly, for WISDM, ASDQR attains 74.95%, exceeding CSKD by 0.41%, CSAG by 1.7%, and KDAG by 14.8%. The results underscore the significance of client selection and adaptive self-knowledge distillation, with CSKD and CSAG consistently outperforming KDAG. Adaptive self-knowledge distillation shows a greater impact on performance than aggregation optimization, while KDAG also contributes positively, highlighting the value of aggregation optimization.

6 Conclusion

In this paper, we present the design, implementation, and evaluation of ASDQR, an efficient and effective FL framework for heterogeneous silos, which significantly achieves communication- and inference-efficient, and inference-effective FL. ASDQR selects high-quality and high-reputation clients for model training, avoiding performance degradation from random client selection and reducing communication costs. Additionally, adaptive local self-knowledge distillation combined with historical knowledge addresses catastrophic challenges of forgetting and improves inference effectiveness, resulting in fewer communication rounds. The aggregation strategy is meticulously crafted to take the quality, reputation, and importance of local model updates into consideration when performing global aggregation for more accurate and effective models. Experimental results conducted on real-world datasets and under various settings demonstrate ASDQR's outperformance in comparison to existing



(a) MNIST (b) CIFAR-10 (c) HAR (d) WISDM

Fig. 11. Performance comparison of ablations over communication rounds when considering 100 clients on MNIST, CIFAR-10, HAR, and WISDM datasets.

works. For future work, our focus will be on developing mechanisms to prevent and detect client misreporting, enhancing the robustness and reliability of FL systems.

Acknowledgment

This work is supported by the National Natural Science Foundation of China under Grant No.92267206 and No. 62032013.

References

[1] Sawsan AbdulRahman, Hanine Tout, Azzam Mourad, and Chamseddine Talhi. 2020. FedMCCS: Multicriteria client selection model for optimal IoT federated learning. *IEEE Internet of Things Journal* 8, 6 (2020), 4723–4735.

[2] Ravikumar Balakrishnan, Tian Li, Tianyi Zhou, Nageen Himayat, Virginia Smith, and Jeff Bilmes. 2022. Diverse client selection for federated learning via submodular maximization. In *International Conference on Learning Representations*.

[3] Jeremy Bernstein, Yu-Xiang Wang, Kamyar Azizzadenesheli, and Animashree Anandkumar. 2018. signSGD: Compressed optimisation for non-convex problems. In *International Conference on Machine Learning*. PMLR, 560–569.

[4] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. 2020. Language models are few-shot learners. *Advances in neural information processing systems* 33 (2020), 1877–1901.

[5] Bin Cao, Long Zhang, Yun Li, Daquan Feng, and Wei Cao. 2019. Intelligent offloading in multi-access edge computing: A state-of-the-art review and framework. *IEEE Communications Magazine* 57, 3 (2019), 56–62.

[6] Yiqiang Chen, Xiaodong Yang, Xin Qin, Han Yu, Biao Chen, and Zhiqi Shen. 2020. Focus: Dealing with label quality disparity in federated learning. *arXiv preprint arXiv:2001.11359* (2020).

[7] Yanyu Cheng, Jianyuan Lu, Dusit Niyato, Biao Lyu, Jiawen Kang, and Shunmin Zhu. 2022. Federated transfer learning with client selection for intrusion detection in mobile edge computing. *IEEE Communications Letters* 26, 3 (2022), 552–556.

[8] Yae Jee Cho, Jianyu Wang, and Gauri Joshi. 2022. Towards understanding biased client selection in federated learning. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 10351–10375.

[9] Yongheng Deng, Feng Lyu, Ju Ren, Yi-Chao Chen, Peng Yang, Yuezhi Zhou, and Yaoxue Zhang. 2021. Fair: Quality-aware federated learning with precise user incentive and model aggregation. In *IEEE INFOCOM 2021-IEEE Conference on Computer Communications*. IEEE, 1–10.

[10] Yongheng Deng, Feng Lyu, Ju Ren, Huaqing Wu, Yuezhi Zhou, Yaoxue Zhang, and Xuemin Shen. 2021. AUCTION: Automated and quality-aware client selection framework for efficient federated learning. *IEEE Transactions on Parallel and Distributed Systems* 33, 8 (2021), 1996–2009.

[11] Enmao Diao, Jie Ding, and Vahid Tarokh. 2020. HeteroFL: Computation and communication efficient federated learning for heterogeneous clients. In *International Conference on Learning Representations*.

[12] Yuting He, Yiqiang Chen, Xiaodong Yang, Yingwei Zhang, and Bixiao Zeng. 2022. Class-wise adaptive self distillation for heterogeneous federated learning. In *Proceedings of the 36th AAAI Conference on Artificial Intelligence, Virtual*, Vol. 22.

[13] Robert Hönig, Yiren Zhao, and Robert Mullins. 2022. DAdaQuant: Doubly-adaptive quantization for communication-efficient Federated Learning. In *International Conference on Machine Learning*. PMLR, 8852–8866.

[14] Tiansheng Huang, Weiwei Lin, Li Shen, Keqin Li, and Albert Y Zomaya. 2022. Stochastic client selection for federated learning with volatile clients. *IEEE Internet of Things Journal* 9, 20 (2022), 20055–20070.

[15] Tiansheng Huang, Weiwei Lin, Wentai Wu, Ligang He, Keqin Li, and Albert Y Zomaya. 2020. An efficiency-boosting client selection scheme for federated learning with fairness guarantee. *IEEE Transactions on Parallel and Distributed Systems* 32, 7 (2020), 1552–1564.

[16] Sohei Itahara, Takayuki Nishio, Masahiro Morikura, and Koji Yamamoto. 2020. Lottery hypothesis based unsupervised pre-training for model compression in federated learning. In *2020 IEEE 92nd Vehicular Technology Conference (VTC2020-Fall)*. IEEE, 1–5.

[17] KR Jayaram, Vinod Muthusamy, Gegi Thomas, Ashish Verma, and Mark Purcell. 2022. Adaptive Aggregation For Federated Learning. *arXiv preprint arXiv:2203.12163* (2022).

[18] Hai Jin, Dongshan Bai, Dezhong Yao, Yutong Dai, Lin Gu, Chen Yu, and Lichao Sun. 2022. Personalized edge intelligence via federated self-knowledge distillation. *IEEE Transactions on Parallel and Distributed Systems* 34, 2 (2022), 567–580.

[19] Jiawen Kang, Zehui Xiong, Dusit Niyato, Shengli Xie, and Junshan Zhang. 2019. Incentive mechanism for reliable federated learning: A joint optimization approach to combining reputation and contract theory. *IEEE Internet of Things Journal* 6, 6 (2019), 10700–10714.

[20] Alex Krizhevsky, Geoffrey Hinton, et al. 2009. Learning multiple layers of features from tiny images. (2009).

[21] Yann LeCun. 1998. The MNIST database of handwritten digits. *http://yann. lecun. com/exdb/mnist/* (1998).

[22] Ang Li, Jingwei Sun, Pengcheng Li, Yu Pu, Hai Li, and Yiran Chen. 2021. Hermes: an efficient federated learning framework for heterogeneous mobile clients. In *Proceedings of the 27th Annual International Conference on Mobile Computing and Networking*. 420–437.

[23] Chenning Li, Xiao Zeng, Mi Zhang, and Zhichao Cao. 2022. PyramidFL: A fine-grained client selection framework for efficient federated learning. In *Proceedings of the 28th Annual International Conference on Mobile Computing And Networking*. 158–171.
[24] Daliang Li and Junpu Wang. 2019. Fedmd: Heterogenous federated learning via model distillation. *arXiv preprint arXiv:1910.03581* (2019).
[25] Qinbin Li, Bingsheng He, and Dawn Song. 2021. Model-contrastive federated learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 10713–10722.
[26] Qinbin Li, Zeyi Wen, Zhaomin Wu, Sixu Hu, Naibo Wang, Yuan Li, Xu Liu, and Bingsheng He. 2021. A survey on federated learning systems: vision, hype and reality for data privacy and protection. *IEEE Transactions on Knowledge and Data Engineering* (2021).
[27] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. 2020. Federated optimization in heterogeneous networks. *Proceedings of Machine learning and systems* 2 (2020), 429–450.
[28] Xiang Li, Kaixuan Huang, Wenhao Yang, Shusen Wang, and Zhihua Zhang. 2019. On the convergence of fedavg on non-iid data. *arXiv preprint arXiv:1907.02189* (2019).
[29] Ying Li, Xingwei Wang, Rongfei Zeng, Mingzhou Yang, Kexin Li, Min Huang, and Schahram Dustdar. 2023. VARF: An Incentive Mechanism of Cross-silo Federated Learning in MEC. *IEEE Internet of Things Journal* (2023).
[30] Weiwei Lin, Yinhai Xu, Bo Liu, Dongdong Li, Tiansheng Huang, and Fang Shi. 2022. Contribution-based Federated Learning client selection. *International Journal of Intelligent Systems* 37, 10 (2022), 7235–7260.
[31] Ji Liu, Juncheng Jia, Hong Zhang, Yuhui Yun, Leye Wang, Yang Zhou, Huaiyu Dai, and Dejing Dou. 2024. Efficient Federated Learning Using Dynamic Update and Adaptive Pruning with Momentum on Shared Server Data. *ACM Transactions on Intelligent Systems and Technology* (2024).
[32] Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Aguera y Arcas. 2017. Communication-efficient learning of deep networks from decentralized data. In *Artificial intelligence and statistics*. PMLR, 1273–1282.
[33] Xiaomin Ouyang, Zhiyuan Xie, Jiayu Zhou, Jianwei Huang, and Guoliang Xing. 2021. Clusterfl: a similarity-aware federated learning system for human activity recognition. In *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*. 54–66.
[34] et al. Reyes-Ortiz, Jorge. 2013. Human Activity Recognition Using Smartphones. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C54S4K.
[35] Felix Sattler, Simon Wiedemann, Klaus-Robert Müller, and Wojciech Samek. 2019. Robust and communication-efficient federated learning from non-iid data. *IEEE transactions on neural networks and learning systems* 31, 9 (2019), 3400–3413.
[36] Shreya Sharma, Chaoping Xing, Yang Liu, and Yan Kang. 2019. Secure and efficient federated transfer learning. In *2019 IEEE international conference on big data (Big Data)*. IEEE, 2569–2576.
[37] Neha Singh, Jatin Rupchandani, and Mainak Adhikari. 2023. Personalized Federated Learning for Heterogeneous Edge Device: Self-Knowledge Distillation Approach. *IEEE Transactions on Consumer Electronics* (2023).
[38] Pengju Wang, Bochao Liu, Dan Zeng, Chenggang Yan, and Shiming Ge. 2023. Personalized Federated Learning via Backbone Self-Distillation. In *Proceedings of the 5th ACM International Conference on Multimedia in Asia*. 1–7.
[39] Tian Wang, Yan Liu, Xi Zheng, Hong-Ning Dai, Weijia Jia, and Mande Xie. 2021. Edge-based communication optimization for distributed federated learning. *IEEE Transactions on Network Science and Engineering* (2021).
[40] Yuwei Wang and Burak Kantarci. 2020. A novel reputation-aware client selection scheme for federated learning within mobile environments. In *2020 IEEE 25th International Workshop on Computer Aided Modeling and Design of Communication Links and Networks (CAMAD)*. IEEE, 1–6.
[41] Yujia Wang, Lu Lin, and Jinghui Chen. 2022. Communication-Efficient Adaptive Federated Learning. *arXiv preprint arXiv:2205.02719* (2022).
[42] Gary Weiss. 2019. WISDM Smartphone and Smartwatch Activity and Biometrics Dataset. UCI Machine Learning Repository. DOI: https://doi.org/10.24432/C5HK59.
[43] Chuhan Wu, Fangzhao Wu, Lingjuan Lyu, Yongfeng Huang, and Xing Xie. 2022. Communication-efficient federated learning via knowledge distillation. *Nature communications* 13, 1 (2022), 1–8.
[44] Jie Xu and Heqiang Wang. 2020. Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective. *IEEE Transactions on Wireless Communications* 20, 2 (2020), 1188–1200.
[45] Miao Yang, Hua Qian, Ximin Wang, Yong Zhou, and Hongbin Zhu. 2021. Client selection for federated learning with label noise. *IEEE Transactions on Vehicular Technology* 71, 2 (2021), 2193–2197.
[46] Hangjia Zhang, Zhijun Xie, Roozbeh Zarei, Tao Wu, and Kewei Chen. 2021. Adaptive client selection in resource constrained federated learning systems: A deep reinforcement learning approach. *IEEE Access* 9 (2021), 98423–98432.
[47] Sai Qian Zhang, Jieyu Lin, and Qi Zhang. 2022. A multi-agent reinforcement learning approach for efficient client selection in federated learning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 36. 9091–9099.
[48] Hongbin Zhu, Yong Zhou, Hua Qian, Yuanming Shi, Xu Chen, and Yang Yang. 2022. Online client selection for asynchronous federated learning with fairness consideration. *IEEE Transactions on Wireless Communications* 22, 4 (2022), 2493–2506.

A Comparative Analysis of Client Selection Methods in FL

Table 3. Comparison Table of Client Selection Methods in FL.

Ref.	Selection Strategy	Communication Cost	Key Contributions	Limitations
[1]	Multicriteria Client Selection	Reduce the number of communication rounds	Require fewer communication rounds and more clients to achieve the desired accuracy	Increase computation and selection overhead.
[2]	Diverse Client Selection	Minimize the communication costs	Using submodular maximization to reduce the variance introduced by CS and improve model accuracy	May introduce variance in client contributions, potentially leading to imbalanced updates.
[7]	All & RL-Based Client Selection	Improve communication efficiency	Integrate RL-based CS with FL and transfer learning	Require high computational overhead and time to converge.
[8]	Power-of-Choice Client Selection	Reduce the number of communication rounds	Flexibly spans between convergence speed and bias	Flexibility may lead to inconsistent convergence or bias issues.
[10]	Automated and Quality-Aware Client Selection	×	Demonstrate the efficiency, robustness, and scalability	Overlook personalization in local models.
[14]	Stochastic Client Selection	×	Optimize CS while addressing the trade-off between participation effectiveness and fairness	Balancing effectiveness and fairness can be challenging.
[30]	Contribution-based Client Selection	×	A good balance between global accuracy and convergence speed	May not address client-specific variability or system constraints.
[40]	Reputation-aware Client Selection	×	An optimal CS method for FL based on reputation scores	Reliance on reputation scores may introduce biases.
[45]	Online Client Selection	×	Find the best client subset based on their relative test performance	Computationally expensive to dynamically select clients.
[46]	Adaptive Client Selection	Increase the number of communication rounds	Energy and training delay required in FL can be reduced	Increased communication rounds can lead to higher latency and costs.
[47]	Efficient run-time Client Selection	Lower communication overhead	Jointly optimizes accuracy, latency and communication efficiency	Lacks robust client weighting based on contribution.
[48]	Online and asynchronous Client Selection	Enhance the utilization of communication resources	Asynchronous FL with a dynamic, fairness-focused CS strategy	Asynchronous methods can complicate fairness and dynamic resource utilization.
Ours	Reputation- and Quality-aware Client Selection	Adaptive self-knowledge distillation	Communication and inference-effective FL	Assume that clients are trustworthy and that communication channels

B Proof of Convergence Analysis

Here, we impose the following definition and assumptions on the functions $L_1, L_2, \cdots, L_n$.

Assumption 1 (L-smooth). $L_1, \cdots, L_n$ are all L-Smooth, *i.e.*, for all v and w, $L_n(v) \leq L_n(w) + (v-w)^T + \frac{L}{2}\|v-w\|_2^2$.

Assumption 2 (μ-strongly Convex). $L_1, \cdots, L_n$ are all μ-strongly convex, *i.e.*, for all v and w, $L_n(v) \geq L_n(w) + (v-w)^T + \frac{\mu}{2}\|v-w\|_2^2$.

Assumption 3 (Bounded Local Variance). For the mini-batch ξ_n uniformly sampled at random from $\mathcal{B}_n$ from client n, the variance of stochastic gradients is bounded: $\mathbb{E}\|\nabla L_n(\omega_n^t, \xi_n^t) - \nabla L_n(\omega_n^t)\| \leq \sigma_n^2$.

Assumption 4 (Bounded Local Gradient). The stochastic gradient's expected squared norm is uniformly bounded, *i.e.*, $\mathbb{E}\|\nabla L_n(\omega_n^t, \xi_n)\|^2 \leq G^2$ for $n = 1, \cdots, N$.

Assumption 5 (Bounded Heterogeneity). $[L^* - \sum_{n\in\mathcal{N}} p_n L_{n^*}]$ denotes the statistical heterogeneity, bounded by W, where $L^* := \min_w L(w)$ and $L_{n^*} := \min_v L_n(v)$.

Definition 1 (Local-Global Objective Gap). For the global optimum $\omega^* = \arg\min_\omega L(\omega)$ and local optimum $\omega^{n^*} = \arg\min_\omega L_n(\omega)$, the local-global objective gap is defined as below:

$$\Gamma \triangleq L^* - \sum_{n=1}^{N} p_n L_{n^*} = \sum_{n=1}^{N} p_n (L_n(\omega^*) - L_n(\omega_{n^*})) \geq 0. \tag{27}$$

where Γ is an inherent property of both the local and global objective functions, independent of the client selection strategy. A larger Γ indicates higher data heterogeneity. If $\Gamma = 0$, it implies that the local and global optimal values align consistently, eliminating solution bias attributed to the client selection strategy.

Theorem 1(Convergence of Algorithm 3). Under Assumptions 1-5 and Definition 1, if choosing $p_n = N_n^c$ (N_c is the normalized contribution of selected clients in Algorithm 2 and the number of local epochs as K, the convergence rate is

$$\begin{aligned} \|\bar{\omega}^{t+1} - \omega^*\|^2 =& \|\bar{\omega}^{t+1} - \bar{v}^{t+1} + \bar{v}^{t+1} - \omega^*\|^2 \\ =& \|\bar{\omega}^{t+1} - \bar{v}^{t+1}\|^2 + \|\bar{v}^{t+1} - \omega^*\|^2 + \\ & 2\langle \bar{\omega}^{t+1} - \bar{v}^{t+1}, \bar{v}^{t+1} - \omega^* \rangle. \end{aligned} \tag{28}$$

(1) If iteration t is not aggregation step, $\bar{\omega}^{t+1} = \bar{v}^{t+1}$ and

$$\|\bar{\omega}^{t+1} - \omega^*\|^2 = \|\bar{v}^{t+1} - \omega^*\|^2. \tag{29}$$

According to **Lemma 1 (Results of one step SGD)** in [28],

$$\begin{aligned} \mathbb{E}\|\bar{v}^{t+1} - \omega^*\|^2 \leq & (1-\eta\mu)\mathbb{E}\|\bar{\omega}^t - \omega^*\|^2 + \eta^2 \mathbb{E}\|g^t - \bar{g}^t\|^2 + \\ & 6L\eta^2\Gamma + 2\mathbb{E}\sum_{n=1}^{n} p_n^t \|\bar{\omega}^t - \omega_n^t\|^2 \\ = & (1-\eta\mu)\mathbb{E}\|\bar{\omega}^t - \omega^*\|^2 + \eta^2 C, \end{aligned} \tag{30}$$

where $C = \sum_{n=1}^{n} (p_n^t)^2 \sigma_n^2 + 6L\Gamma + 8(K-1)^2 G^2$.

(2) If iteration t is the aggregation step,

$$\begin{aligned} \mathbb{E}[\|\bar{\omega}^{t+1} - \omega^*\|]^2 \leq & (1-\eta\mu)\mathbb{E}[\|\bar{v}^{t+1} - \omega^*\|^2] + KG\Upsilon\eta + \\ & [(LG\Upsilon + (LGK\eta + G)^2)K^2 + B]\eta^2 \end{aligned} \tag{31}$$

where $\mathbb{E}[\|\bar{v}^{t+1} - \omega^*\|] \leq \Upsilon$. According to Assumptions 2 and 5, $\left\|\sum_{n\in\mathcal{N}} p_n {v_n}^* - \omega^*\right\|$ is bounded by a constant U, as shown in Eq. (32).

$$\begin{aligned}\left\|\sum_{n\in\mathcal{N}} p_n {v_n}^* - \omega^*\right\| &\leq \sum_{n\in\mathcal{N}} p_n \left\|\omega^* - {v_n}^*\right\| \\ &\leq \sum_{n\in\mathcal{N}} p_n (1 + \left\|\omega^* - {v_n}^*\right\|^2) \\ &\leq 1 + \frac{2}{\mu}\left(L^* - \sum_{n\in\mathcal{N}} p_n L_n^*\right) \\ &\leq 1 + \frac{2W}{\mu} = U\end{aligned} \tag{32}$$

where $w^* = \arg\min_w L(w)$ and ${v_n}^* = \arg\min_v L_n(v)$ for $n \in \mathcal{N}$.

Below is the convergence analysis. Here, we assume the selected clients perform τ time steps of local update, the selected set $\mathcal{N}_s^t$ remains constant for every τ iteration. Namely, if $(t+1)\ mod\ \tau = 0$, then $\mathcal{N}_s^{t+1} = \mathcal{N}_s^{t+2} = \cdots = \mathcal{N}_s^{t+\tau}$.

$$v_n^{t+1} = \omega_n^t - \eta \nabla L_n(\omega_n^t; x_n^t). \tag{33}$$

$$\omega_n^{t+1} = \begin{cases} v_n^{t+1}, & for\ (t+1)\ mod\ \tau \neq 0, \\ \frac{1}{N_s} \sum\limits_{j\in\mathcal{N}_s} \left(\omega_j^t - \eta \nabla L_j(\omega_j^t; x_j^t)\right) \triangleq \bar{\omega}^{t+1}, & otherwise. \end{cases} \tag{34}$$

Let

$$\bar{v}^t := \sum_{n\in\mathcal{N}} p_n^t v_n^t. \tag{35}$$

$$\bar{\omega}^t := \sum_{n\in\mathcal{N}} p_n^t \omega_n^t. \tag{36}$$

where p_n^t is the weight of n^{th} client and $\sum\limits_n p_n^t = 1$. Similar with $\mathcal{N}_s^t$, $p_n^{t+1} = p_n^{t+2} = \cdots = p_n^{t+\tau}$ when $(t+1)\ mod\ \tau = 0$. Therefore,

$$\bar{\omega}^t = \begin{cases} \bar{v}^t, & for\ (t+1)\ mod\ \tau \neq 0, \\ \frac{1}{N_s} \sum\limits_{j\in\mathcal{N}_s} v_j^t, & otherwise, \end{cases} \tag{37}$$

and,

$$\bar{v}^{t+1} = \bar{\omega}^t - \eta \left(\sum_{n\in\mathcal{N}} p_n^t L_n(\omega_n^t; x_n^t)\right) := \bar{\omega}^t - \eta g^t. \tag{38}$$

We have

$$\begin{aligned}\left\|\bar{\omega}^{t+1} - \omega^*\right\|^2 &= \left\|\bar{\omega}^{t+1} - \bar{v}^{t+1} + \bar{v}^{t+1} - \omega^*\right\|^2 \\ &= \underbrace{\left\|\bar{\omega}^{t+1} - \bar{v}^{t+1}\right\|^2}_{A1} + \underbrace{\left\|\bar{v}^{t+1} - \omega^*\right\|^2}_{A2} + \underbrace{2\langle \bar{\omega}^{t+1} - \bar{v}^{t+1}, \bar{v}^{t+1} - \omega^* \rangle}_{A3}.\end{aligned} \tag{39}$$

(1) When $(t+1)\ mod\ \tau \neq 0$,

$$\bar{\omega}^{t+1} = \bar{v}^{t+1}, \tag{40}$$

Hence,

$$\|\bar{\omega}^{t+1} - \omega^*\|^2 = \|\bar{v}^{t+1} - \omega^*\|^2. \tag{41}$$

According to **Lemma 1 (Results of one step SGD)** [28],

$$\begin{aligned}\mathbb{E}\|\bar{v}^{t+1} - \omega^*\|^2 &\leq (1-\eta\mu)\mathbb{E}\|\bar{\omega}^t - \omega^*\|^2 + \eta^2\mathbb{E}\|g^t - \bar{g}^t\|^2 + 6L\eta^2\Gamma + 2\mathbb{E}\sum_{n=1}^{n} p_n^t\|\bar{\omega}^t - \omega_n^t\|^2 \\ &= (1-\eta\mu)\mathbb{E}\|\bar{\omega}^t - \omega^*\|^2 + \eta^2 C,\end{aligned} \tag{42}$$

where $C = \sum_{n=1}^{n} (p_n^t)^2\sigma_n^2 + 6L\Gamma + 8(K-1)^2G^2$.

(2) When $(t+1)\ mod\ \tau = 0$, we need to bound

$$\mathbb{E}[\|\bar{\omega}^{t+1} - \omega^*\|^2] = \mathbb{E}[A1] + \mathbb{E}[A2] + \mathbb{E}[A3], \tag{43}$$

We assume the last time of aggregation is at step $t' = t + 1 - K$ and $\mathcal{N}^s$ is the selected subset of t while $\mathcal{N}^{s'}$ is the selected subset of t' ($\mathcal{N}^s \subseteq \mathcal{N}^{s'}$). To bound the first term $A1$,

$$\begin{aligned}\|\bar{\omega}^{t+1} - \bar{v}^{t+1}\| &= \left\|\left(\bar{\omega}_{t'} + \frac{1}{N_s}\sum_{n\in\mathcal{N}_s}\sum_{\tau=t'}^{t}\Delta v_n^\tau\right) - \left(\bar{\omega}_{t'} + \frac{1}{N_{s'}}\sum_{n'\in\mathcal{N}^{s'}}\sum_{\tau=t'}^{t}\Delta v_{n'}^\tau\right)\right\| \\ &= \left\|\sum_{\tau=t'}^{t}\left(\frac{1}{N_s}\sum_{n\in\mathcal{N}_s}\Delta v_n^\tau - \frac{1}{N^{s'}}\sum_{n'\in\mathcal{N}^{s'}}\Delta v_{n'}^\tau\right)\right\| \\ &\leq \sum_{\tau=t'}^{t}\left\|\left(\frac{1}{N_s}\sum_{n\in\mathcal{N}_s}\Delta v_n^\tau - \frac{1}{N^{s'}}\sum_{n'\in\mathcal{N}^{s'}}\Delta v_{n'}^\tau\right)\right\| \\ &= \sum_{\tau=t'}^{t}\eta\left\|\left(\frac{1}{N_s}\sum_{n\in\mathcal{N}_s}\nabla L_n(v_n^\tau) - \frac{1}{N^{s'}}\sum_{n'\in\mathcal{N}^{s'}}\nabla L_n(v_{n'}^\tau)\right)\right\|,\end{aligned} \tag{44}$$

where

$$\begin{aligned}\left\|\left(\frac{1}{N_s}\sum_{n\in\mathcal{N}_s}\nabla L_n(v_n^\tau) - \frac{1}{N^{s'}}\sum_{n'\in\mathcal{N}^{s'}}\nabla L_n(v_{n'}^\tau)\right)\right\| \leq &\underbrace{\left\|\left(\frac{1}{N_s}\sum_{n\in\mathcal{N}_s}\nabla L_n(v_n^\tau) - \frac{1}{N_s}\sum_{n\in\mathcal{N}_s}\nabla L_n(v_n^{t'})\right)\right\|}_{a1} + \\ &\underbrace{\left\|\left(\frac{1}{N_s}\sum_{n\in\mathcal{N}_s}\nabla L_n(v_n^{t'}) - \frac{1}{N^{s'}}\sum_{n\in\mathcal{N}^{s'}}\nabla L_n(v_n^{t'})\right)\right\|}_{a2} + \\ &\underbrace{\left\|\left(\frac{1}{N^{s'}}\sum_{n\in\mathcal{N}^{s'}}\nabla L_n(v_n^\tau) - \frac{1}{N^{s'}}\sum_{n\in\mathcal{N}^{s'}}\nabla L_n(v_n^{t'})\right)\right\|}_{a3},\end{aligned} \tag{45}$$

According to Assumption 1 (L-smooth of $L_n(\cdot)$) and Assumption 4 (G-bound norm of its stochastic gradient), we have

$$a1 \leq LG \sum_{i=t'}^{\tau} \eta_i, \tag{46}$$

and

$$a3 \leq LG \sum_{i=t'}^{\tau} \eta_i, \tag{47}$$

Next,

$$\begin{aligned} a2 &= \left\| \left(\frac{1}{N_s} \sum_{n \in \mathcal{N}_s} \nabla L_n(v_n^{t'}) - \frac{1}{N^{s'}} \sum_{n \in \mathcal{N}^{s'}} \nabla L_n(v_n^{t'}) \right) \right\| \\ &= \left\| \left(\frac{1}{N_s} \sum_{n \in \mathcal{N}_s} \nabla L_n(v_n^{t'}) - \frac{1}{N_s} \sum_{n \in \mathcal{N}_s} \nabla L_n(v_n^{t'}) - \frac{1}{N^{s'} - N_s} \sum_{n' \in \mathcal{N}^{s'} \setminus \mathcal{N}^s} \nabla L_n(v_{n'}^{\tau}) \right) \right\| \\ &= \left\| \frac{1}{N^{s'} - N_s} \sum_{n' \in \mathcal{N}^{s'} \setminus \mathcal{N}^s} \nabla L^{n'}(v_{n'}^{\tau}) \right\| \\ &\leq G, \end{aligned} \tag{48}$$

Thus,

$$\begin{aligned} \|\bar{\omega}^{t+1} - \bar{v}^{t+1}\| &\leq \sum_{\tau=t'}^{t} \eta \left\| \left(\frac{1}{N_s} \sum_{n \in \mathcal{N}_s} \nabla L_n(v_n^{\tau}) - \frac{1}{N^{s'}} \sum_{n' \in \mathcal{N}^{s'}} \nabla L_n(v_{n'}^{\tau}) \right) \right\| \\ &\leq 2LG \sum_{\tau=t'}^{t} \sum_{i=t'}^{\tau} \eta_\tau \eta_i + KG\eta_\tau \\ &\leq LGK(K-1)\eta_{t'}^2 + KG\eta_{t'} \end{aligned} \tag{49}$$

In this work, we set the learning rate η as fixed. Thus, $\|\bar{\omega}^{t+1} - \bar{v}^{t+1}\| \leq LGK(K-1)\eta^2 + KG\eta$.

Therefore, Eq. (33) can be bounded as follows:

$$\begin{aligned} \mathbb{E}[\|\bar{\omega}^{t+1} - \omega^*\|]^2 &\leq \mathbb{E}[\|\bar{\omega}^{t+1} - \bar{v}^{t+1}\|^2] + \mathbb{E}[\|\bar{v}^{t+1} - \omega^*\|^2] + 2\mathbb{E}[\langle \bar{\omega}^{t+1} - \bar{v}^{t+1}, \bar{v}^{t+1} - \omega^* \rangle] \\ &\leq (LGK(K-1)\eta^2 + KG\eta)^2 + [(1-\eta\mu)\mathbb{E}[\|\bar{\omega}^t - \omega^*\|^2] + \eta^2 C] + 2(GK(K-1)\eta^2 + KG\eta)\mathbb{E}[\|\bar{v}^{t+1} - \omega^*\|] \\ &\leq (1-\eta\mu)\mathbb{E}[\|\bar{v}^{t+1} - \omega^*\|^2] + KG\Upsilon\eta + [B + LGK(K-1)\Upsilon + (LGK(K-1)\eta + KG)^2]\eta^2 \\ &\leq (1-\eta\mu)\mathbb{E}[\|\bar{v}^{t+1} - \omega^*\|^2] + KG\Upsilon\eta + [(LG\Upsilon + (LGK\eta + G)^2)K^2 + B]\eta^2 \end{aligned} \tag{50}$$

where $\mathbb{E}[\|\bar{v}^{t+1} - \omega^*\|] \leq \Upsilon$.

$$
\begin{aligned}
\mathbb{E}[\left\|\bar{v}^{t+1}-\omega^{*}\right\|] &\leq \mathbb{E}\left\|\bar{v}^{t+1}-\sum_{n\in\mathcal{N}} p_n {v_n}^{*}\right\| + \mathbb{E}\left\|\sum_{n\in\mathcal{N}} p_n {v_n}^{*}-\omega^{*}\right\| \\
&\leq \mathbb{E}\left\|\bar{v}^{t+1}-\sum_{n\in\mathcal{N}} p_n {v_n}^{*}\right\| + U \\
&\leq \sum_{n\in\mathcal{N}} \mathbb{E}\|p_n(\bar{v}^{t+1}-{v_n}^{*})\| + U \\
&\leq \sum_{n\in\mathcal{N}} \frac{p_n}{\mu}\mathbb{E}\left\|\nabla L_n(v_n^t)\right\| + U \\
&\leq \frac{G}{\mu} + U \\
&\leq \Upsilon
\end{aligned}
\tag{51}
$$

Here the proof is completed.

C Buffer Performance Tables

Table 4. Inference Accuracy Overview Under Non-IID(MLP).

	Buffer1	Buffer3	Buffer5	Buffer7	Buffer9	Buffer10	Buffer11	Buffer13	Buffer15	ABuffer5	ABuffer10	ABuffer15
Round 50	80.69	80.32	80.80	80.80	80.80	80.13	80.52	80.29	72.25	81.00	80.20	80.86
Round 100	82.25	81.81	82.29	82.20	82.10	81.70	81.70	81.73	75.32	80.76	81.56	82.70
Round 150	82.37	81.92	82.32	82.27	82.24	81.74	81.71	81.71	77.68	80.74	81.76	82.95
Round 200	82.32	81.83	82.38	82.21	82.04	81.80	81.76	81.95	78.92	80.36	82.54	82.98

Table 5. Inference Accuracy Overview Under Non-IID(CNN).

	Buffer1	Buffer3	Buffer5	Buffer7	Buffer9	Buffer10	Buffer11	Buffer13	Buffer15	ABuffer5	ABuffer10	ABuffer15
Round 50	87.68	87.87	88.60	87.68	87.63	87.98	87.55	82.00	74.04	81.90	87.79	88.24
Round 100	89.65	90.80	90.69	90.40	90.88	89.94	90.76	85.74	77.32	82.80	91.48	91.33
Round 150	90.19	91.10	91.39	90.81	91.39	90.27	91.38	88.14	83.77	84.19	91.81	92.04
Round 200	90.29	91.19	91.50	91.14	91.76	90.57	91.52	89.18	84.69	85.57	92.05	92.39

Table 6. Inference Accuracy Overview Under Non-IID(ResNet18).

	Buffer1	Buffer3	Buffer5	Buffer7	Buffer9	Buffer10	Buffer11	Buffer13	Buffer15	ABuffer5	ABuffer10	ABuffer15
Round 50	93.46	95.39	94.96	94.51	94.93	94.93	94.67	95.17	94.74	94.64	95.00	95.08
Round 100	93.69	95.25	94.75	94.57	94.64	95.14	94.68	95.10	94.89	94.46	95.32	95.07
Round 150	93.58	95.24	94.95	94.63	95.04	95.21	94.71	95.02	94.86	94.32	95.12	95.24
Round 200	93.42	95.19	94.71	94.49	94.90	95.27	94.64	95.07	94.86	94.37	95.27	95.33

D Client Buffer Selection and Update

Algorithms 4 and 5 create a robust mechanism for managing client buffers in resource-constrained environments. Algorithm 4 optimizes buffer selection based on system resources, while Algorithm 5 ensures efficient and dynamic management of the buffer during training. This combination supports scalable and adaptive operations in federated learning scenarios, particularly under heterogeneous client conditions.

Algorithm 4 Client Buffer Selection.

Input: resources dict: $resources\_n$, HighMemoryThreshold: hmt, MediumMemoryThreshold: mmt, LowCPUThreshold: lct, MediumCPUThreshold: mct.
Output: buffer_type, buffer_size or b.

function ClientBufferSelection($resources\_n$)
  $Fixed\_buffer\_options \leftarrow [1, 5, 10, 15]$
  $Adaptive\_b\_options \leftarrow [1, 5, 10, 15]$
  if $memory\_availability >= hmt \quad and \quad cpu\_usage < lct$ **then**
    buffer_type $\leftarrow$ "$adaptive$"
    b $\leftarrow Adaptive\_b\_options[2]$
  else if $memory\_availability >= mmt \quad and \quad cpu\_usage < mct$ **then**
    buffer_type $\leftarrow$ "$fixed$"
    buffer_size $\leftarrow Fixed\_buffer\_options[1]$
  else
    buffer_type $\leftarrow$ "$fixed$"
    buffer_size $\leftarrow Fixed\_buffer\_options[0]$
  end if
  return buffer_type, buffer_size or b
end function

Algorithm 5 Update Buffer.

Input: $buffer_n, w_n^{t_c,K}, buffer\_type, t_c, fixed\_size, b$.
Output: $buffer_n$.

function UpdateBuffer($buffer_n, w_n^{t_c,K}, buffer\_type, t_c, fixed\_size, b$)
  $B^t \leftarrow \lceil t_c/b \rceil$ ▷ Adaptive size
  if $buffer\_type =$ "$fixed$" **then**
    $buffer\_size \leftarrow fixed\_size$
  else
    $buffer\_size \leftarrow B^t$
  end if
  if $size(buffer_n) \geq buffer\_size$ **then**
    $buffer_n \leftarrow buffer_n[1:]$
  end if
  $buffer_n \leftarrow buffer_n + \{w_n^{t_c,K}\}$
end function

E Communication Efficiency Guarantee

To prove the communication efficiency guarantee of the proposed ASDQR framework, we need to analyze the communication costs associated with client selection and local model updates under the designed mechanisms. Specifically, we will conduct a quantitative analysis of the communication efficiency from the following aspects:

(1) Communication Overhead for Client Selection
In each round t, the client selection process ensures that only a subset of clients $\mathcal{N}_s$ is selected based on quality and reputation estimates. The selection criterion is based on a composite quality score, which combines several factors such as loss disparity, data volume, label distribution, and distribution differences. Let N_s represent the number of clients selected in each round. Let N be the total number of clients in the system. The communication overhead is proportional to the number of clients selected in each round. Since the contribution of each client is evaluated using the formula in Eq. (16), we can conclude that only the clients with the highest contribution will be selected, thereby reducing the total communication.
Thus, the communication overhead for client selection is $O(N_s)$, where N_s is expected to be much smaller than the total number of clients N. This reduces communication costs, as fewer clients are involved in transmitting updates, thereby avoiding redundant data transmissions.

(2) Communication Overhead for Model Updates
Each selected client n sends its local model update ω_n^t in round t, but the size of these updates can be reduced by employing the knowledge distillation mechanism.
Without Knowledge Distillation: Each client sends the full model update to the server. The communication cost for a client update is proportional to the size of the model, denoted by $|\omega_n^t|$. If N_s clients are selected, the total communication cost is:

$$Total\_Communication\_Cost(NoDistillation) = O(N \cdot |\omega_n^t|) \tag{52}$$

With Knowledge Distillation (Scheme 5 and Scheme 6): The local model update is regularized by a distillation term. In Scheme 5, the local guiding model $\hat{\omega}_n$ is the average of historical models over the last B^t rounds. The size of the local model update is reduced due to the distillation process, as only the "distilled knowledge" from previous rounds is communicated. The communication cost is thus reduced to:

$$Total\_Communication\_Cost(WithDistillation) = O(N \cdot |\omega_n^{t_c}| \cdot \lambda) \tag{53}$$

where λ controls the weight of distillation, reducing the size of the model in each round, which further lowers communication overhead.

(3) Buffer-Adaptive Scheme and Communication Overhead In the buffer-adaptive scheme, the buffer size B^t controls the amount of historical knowledge used for distillation in each round. As B^t increases over time, clients only transmit a portion of their models, effectively reducing the total amount of data exchanged. The communication overhead for the buffer-adaptive scheme is:

$$Total\_Communication\_Cost(Buffer\_Adaptive) = O(N \cdot |\omega_n^{t_c}| \cdot B^t) \tag{54}$$

By using the buffer-adaptive mechanism, the number of model updates transmitted is minimized, leading to a significant reduction in communication costs. This also enables a more efficient use of bandwidth over multiple communication rounds.

(4) Communication Efficiency Guarantee
The communication efficiency guarantee can be expressed as:

$$Communication\_Efficiency = O(\frac{N \cdot |\omega_n^{t_c}| \cdot B^t \cdot \lambda}{N_s}) \tag{55}$$

where N is the total number of clients. $|\omega_n^{t_c}|$ is the size of the distilled model update. B^t is the buffer size for historical knowledge, for clients with limited resources, a small fixed buffer in Appendix D (e.g., Buffer1) is more suitable, as it reduces computational and storage overhead. λ is the distillation weight. N_s is the number of clients selected per round. Since N_s is much smaller than N(as shown in Figure 6, $\frac{N_s}{N}$, converging to 0.13-0.2 at the 100th communication round), and the model updates are reduced in size through distillation and historical knowledge buffers, the communication efficiency is significantly improved.

F Computational Complexity Analysis

To address the concerns regarding the computational overhead on both the client and server sides for the ASDQR framework, we can provide an analysis of the respective computational costs for each process involved.

(1) Client-Side Computational Overhead.
On the client side, the ASDQR framework requires clients to participate in three key processes: local model training, reputation/quality scoring, and self-knowledge distillation. Below is an estimated breakdown of the computational cost for each process:
   (a) **Local Model Training:** Depends on the size of the dataset, model complexity, and number of local epochs. For models like MLP, CNN and ResNet18, the complexity is $O(x_n \times K \times p)$, where x_n is the number of local samples, K is the number of local epochs, and p is the number of parameters in the updated model.
   (b) **Reputation and Quality Scoring:**Reputation scoring involves lightweight calculations such as model accuracy evaluation and update contribution measurements, which are typically $O(x_n)$.
   (c) **Self-Knowledge Distillation:** Self-distillation is performed after local training and depends on the number of past models stored for distillation.
   For a historical sequence of k models, the cost of distilling knowledge can be approximated as $O(k \times x_n \times p)$. The distillation can be parallelized, and the value of k can be dynamically adjusted based on client resources.

   Total Computation on Clients: The overall client-side computational cost is dominated by the local training and distillation processes. Since distillation occurs after training, it can be made adaptive (adjustable based on resource availability). Clients with limited computational resources can reduce the depth of self-distillation (lower k) or perform fewer local epochs, ensuring the framework is scalable across heterogeneous clients.

(2) Server-Side Computational Overhead.
On the server side, the ASDQR framework handles client selection and model aggregation. The computational costs associated with these processes are generally lightweight compared to the client-side operations.
   (a) **Client Selection:** The server selects clients for the next round based on their reputation and quality scores. This selection process involves sorting and filtering clients based on their scores. If there are N clients, sorting them based on reputation requires $O(NlogN)$ operations. Since this is performed once per communication round, it does not impose significant overhead.
   (b) **Model Aggregation:** After receiving model updates from clients, the server performs weighted model aggregation. The weights are determined based on clients' contribution and reliability, the cost of model aggregation is $O(N * p)$.

   Total Computation on the Server: The overall server-side computational overhead remains low due to the lightweight nature of client selection and model aggregation tasks. Since the heavy lifting (i.e., model training) is performed by clients, the server operations are computationally manageable.

(3) Computational Cost Analysis Table.

Process	**Computational Complexity**	**Comments**
Client-Side		
Local Model Training	$O(x_n \times K \times p)$	Training depends on data size, local epochs, and model update parameters.
Reputation and Quality Scoring	$O(x_n)$	Lightweight process based on model evaluation (minimal overhead).
Self-Knowledge Distillation	$O(k \times x_n \times p)$	Distillation depends on historical model count (k) and is scalable.
Server-Side		
Client Selection	$O(NlogN)$	Sorting of client reputations for selection (lightweight compared to training).
Model Aggregation	$O(N \times p)$	Weight-based model aggregation (parallelizable, but dependent on model parameters p).

(4) **Real-World Scalability Considerations**.

(a) Client-Side Scalability: The ASDQR framework is designed to be adaptive, allowing clients to adjust their self-knowledge distillation depth based on their resources. Clients with more powerful hardware can use a higher k for better model personalization, while resource-constrained clients can reduce k to minimize overhead.

(b) Server-Side Scalability: The computational overhead on the server is lightweight relative to the client-side operations, making ASDQR suitable for deployment in real-world edge computing scenarios. The model and client aggregation processes can be easily parallelized using modern server infrastructure, ensuring that the server is not a bottleneck in the system.

Received 25 September 2024; revised 22 December 2024; accepted 1 February 2025