

Resilient Federated Learning on Embedded Devices with Constrained Network Connectivity

Zihan Li[†], Han Liu[†], Ao Li[†], Ching-hsiang Chan[†],
Yevgeniy Vorobeychik[†], William Yeoh[†], Wenjing Lou[‡], Ning Zhang[†]

[†] Washington University in St. Louis, MO, USA

[‡] Virginia Polytechnic Institute and State University, VA, USA

{tomson.li, h.liu1, ao, c.ching-hsiang, yvorobeychik, wyeoh, zhang.ning}@wustl.edu, wjlou@vt.edu

Abstract—Federated learning enables decentralized model training while preserving data privacy. However, since the learning process overlays the physical network infrastructure, the efficiency of learning can be impacted by network connectivity. In this work, we conducted extensive experiments to empirically characterize the impacts and leverage the insights to propose an adaptive federation framework, where clients with limited bandwidth are only prompted to transmit adaptively compressed gradient updates when the gradient similarity score is similar between the local and global models. Our evaluation in simulated environments and on real hardware devices shows bandwidth savings of 60% to 78% compared to state-of-the-art methods.

I. INTRODUCTION

Federated learning (FL) has emerged as a transformative paradigm in decentralized machine learning, allowing distributed devices to collaboratively train a global model while preserving the privacy of local data [1]. This approach has gained widespread attention for its applications in privacy-sensitive domains such as healthcare [2], [3], finance [4], smart devices [5], and personalized services [6], [7]. Federated learning operates as a distributed system of nodes with varying computing power and network connectivity. As an overlay on existing network infrastructure, its success heavily depends on underlying network capabilities.

Existing Solutions and Limitations: Recognizing this challenge, existing works have investigated latency or bandwidth constraint optimization techniques. Protocol-level optimizations [8], [9] focus on timing alignment to mitigate staleness issues due to network latency, whereas model-level techniques [10], [11] focus on reducing the size of the gradient for transmission due to bandwidth constraints. However, all of these strategies are static, which are pre-defined according to the target execution environment and network conditions. In contrast, real-world network environments are often highly dynamic, requiring strategies that can adapt to real-time changes in network connectivity.

Examination of FL Resiliency: To understand the challenges and opportunities, we conducted an empirical study of FL under different network conditions and varying model complexities. Our investigation revealed two key insights that hold valid across different datasets, models, data distributions, and both synchronous and asynchronous FL settings: (1) A moderate level of client update dropouts (approximately 20%)

has minimal adverse effects on global model accuracy. (2) In asynchronous FL, model staleness poses an even more negative impact on accuracy compared to client dropouts.

Our Solution – AdaFL: This paper proposes AdaFL, a novel adaptive FL framework that operates in real-time to enhance communication efficiency while maintaining model performance. Based on our first insight, we propose an adaptive node selection algorithm to optimize on client selection. Instead of requesting updates from all available clients, AdaFL adaptively evaluates the utility of each update on the global model’s learning trajectory and selectively chooses updates from clients with meaningful updates and sufficient network bandwidth. This approach minimizes communication overhead by prioritizing impactful updates while preserving model performance. Based on our second insight, we conclude that a timely model update from server to client is crucial for model accuracy. Hence, to ensure the timeliness of the update, we propose to adjust the gradient compression rate dynamically. This approach minimizes unnecessary communication overhead for clients with less impactful updates and more constrained bandwidth while preserving sufficient information from clients that provide meaningful contributions, allowing both timely and informational model updates.

Implementation and Evaluation: We have conducted an extensive evaluation of AdaFL across different datasets and models. The evaluation results demonstrate that AdaFL outperforms state-of-the-art (SOTA) FL methods up to 30% on testing accuracy while reducing 60% to 78% communication cost. Moreover, AdaFL incurs a negligible computational overhead, with an average expansion of 0.05% CPU cycles.

Contributions: This paper makes the following contributions:

- **Empirical Study of FL Resilience:** We empirically showed that FL can remain robust under asynchronous updates and client dropouts. This finding underscores the opportunity for optimizing communication without compromising accuracy.
- **Design of AdaFL:** We propose a novel FL framework that selectively aggregates client updates and adapts the clients’ gradient compression rates based on the utility of their contributions and their network connectivity.
- **Evaluation:** We demonstrate that AdaFL outperforms state-of-the-art (SOTA) FL methods.

II. RELATED WORK

Existing works generally address network constraints from either protocol-level or model-level perspectives.

Protocol-level: Existing protocol optimizations primarily target reducing wait time from network latency. Zhu et al. [8] parallelize communication and training, delaying server aggregation such that client-side wait time for *stragglers* is minimized. Zhang et al. [9] reduces server aggregation blocking, enabling global model updates during aggregation. FedAT [12] introduces a tiering module that groups clients by network performance for tier-based aggregation. However, these approaches focus solely on improving latency but often neglect model performance implications. In practice, reduced latency doesn't always translate to better model quality. AdaFL addresses this limitation by jointly optimizing both objectives—minimizing latency while maintaining model performance.

Model-level: At the model level, reducing transmitted data is a common strategy, achieved through gradient quantization [11], [13] or compression [10], [14], [15]. Another approach encodes local models into lower-dimensional representations to minimize transmission overhead [16]. Xiong et al. [17] replace full model updates with synthesized datasets, while Wang et al. [18] use progressive learning to shrink model size. Though these methods mitigate communication bottlenecks to some extent, they often assume the network condition is static, whereas, in practice, network conditions are highly dynamic.

III. EXPLORATION OF FL NETWORK RESILIENCY

A. Modeling Impacting Factors on FL

In FL, client-server interactions involve downloading the global model and uploading local updates. In each communication round, clients download the global model from the server and perform model training. After local training, clients upload their updated local models to the server for aggregation. Varying network conditions can introduce latency on the uplink, downlink, or both.

We formalize the impact of network conditions on FL performance by analyzing its underlying optimization process. The optimization problem in FL is:

$$\min_w F(w) = \sum_{i=1}^N p_i F_i(w), \quad (1)$$

where $w \in \mathbb{R}^d$ represents the global model parameters, N is the total number of participating clients, $F_i(w)$ is the local loss function for client i , and $p_i \geq 0$ is the weighting coefficient assigned to client i . Typically, in synchronous settings, p_i is equal to w_i , which is defined as $w_i = n_i/n$, where n_i denotes the number of data samples held by client i , and $n = \sum_{i=1}^N n_i$ represents the total number of data samples across all clients.

During each communication round n , client i computes and sends its local gradient g_i^n to the server. The server then performs aggregation using the following equation:

$$G^{n+1} = G^n + \sum_{i=1}^N p_i g_i^n, \quad (2)$$

where G^n is the global gradient at round n , and G^{n+1} is the updated global gradient after aggregation.

To assess the impact of network delay, we analyze two scenarios: synchronous FL and asynchronous FL. Previous research has predominantly focused on synchronous FL [19], [20], [21], where the server waits for all clients to submit their local updates before aggregating them to update the global model. Let Ψ_i denote the computation time for client i , while Υ_i^u and Υ_i^d represent the uplink and downlink network delays for client i , respectively. Under the synchronous protocol, the total time required for one global iteration is:

$$T_{\text{sync}} = \max_i (\Psi_i + \Upsilon_i^u + \Upsilon_i^d). \quad (3)$$

Thus, the synchronization time is constrained by the slowest client due to network delays. Furthermore, if an update from a particular client fails to reach the server, e.g., in cases of unreliable connections, the server may indefinitely wait for this update. The server can instead impose a maximum wait time, dropping any delayed updates beyond this threshold; however, this approach may reduce the accuracy of the global model.

For asynchronous FL, we denote T_i^n as the total time including the training time for performing its local work and sending it to the server aggregation, thus we have $T_i^n = \Psi_{i,n} + \Upsilon_{i,n}^u + \Upsilon_{i,n}^d$, we assume a typical asynchronous setting where the server will wait for Δt^n to perform the aggregation [22], [23]. Note that if the server sets the round time $\Delta t^n = \max_i T_i^n$, the aggregation occurs after receiving contributions from all clients; on the client, if $\Delta t^n = \min_i T_i^n$, it will become fully asynchronous, where every client update results in a server model update.

Comparing T_i^n with Δt^n indicates whether a client is participating in the optimization round or not. When $\mathbb{I}(T_i^n \leq \Delta t^n) = 1$, the local work of client i is used to create the new global model θ^{n+1} , while client i does not contribute when $\mathbb{I}(T_i^n \leq \Delta t^n) = 0$. To account for their scenarios of asynchronous updates, we introduce the aggregation weight $d_i(n)$ corresponding to the weight given by the server to client i at optimization round n , we can then denote it as:

$$p_i(n) = \mathbb{I}(T_i^n \leq \Delta t^n) w_i, \quad (4)$$

where w_i follows the same definition as in synchronous FL, generally referring to the proportion of data samples owned by client i . Under asynchronous aggregation, the global update is performed using the following scheme:

$$G^{n+1} = G^n + \sum_{i=1}^N \mathbb{I}(T_i^n \leq \Delta t^n) p_i g_i^n. \quad (5)$$

From the above analysis, network conditions primarily influence the client dropout rate in synchronous settings and the staleness of client updates in asynchronous settings.

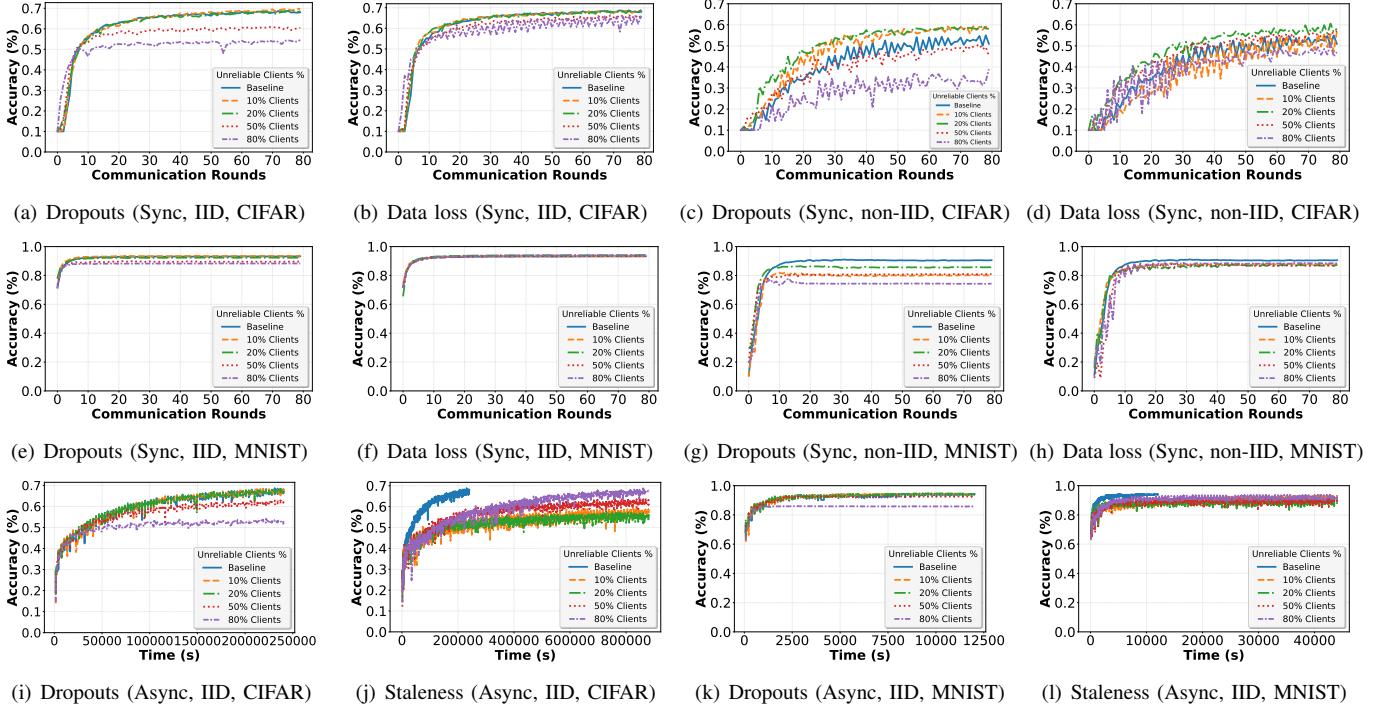


Fig. 1. Testing accuracy of ResNet-50 on CIFAR-10 and CNN on MNIST under different network conditions, data distributions, and FL protocols.

B. Empirical Measurement of Network Impact on FL

To understand how these factors influence client-server interactions and the overall training process, we simulated diverse link latencies and bandwidth constraints. Our examination covered both synchronous and asynchronous FL protocols to analyze their behavior under challenging network conditions. Additionally, we compared performance across different data distribution settings, including IID (Independent and identically distributed) and non-IID scenarios, to highlight how data heterogeneity interacts with network dynamics.

Synchronous and Asynchronous FL: The impact of network conditions on FL performance differs between synchronous and asynchronous protocols. In synchronous FL, high latency causes intermittent client updates to reach the server (*data loss*), disrupting aggregation and delaying global model updates. In asynchronous FL, high latency exacerbates staleness, where client updates are based on outdated global model versions (*staleness*). In our experimental setup, high-latency clients in synchronous FL update the server every other communication round, while in asynchronous settings, they update at a rate 3× slower than other clients, contributing only after others have updated three times.

Network Conditions: We primarily emulate various networks from the perspectives of bandwidth and link conditions.

Bandwidth: The effect of link bandwidth is consistent across both protocols: insufficient bandwidth prevents clients from transmitting updates (*dropouts*), reducing data diversity and contributing to the global model.

Link Conditions: We consider uplink and downlink collectively, as their impact on model updates is equivalent. Specif-

ically, we examine two network conditions: link latency and link bandwidth. High latency delays client updates, slowing the server's aggregation process. Insufficient bandwidth prevents clients from transmitting updates, excluding their contributions from training. We use simulated network data from the ns3 network simulator [24] to model suboptimal network links for clients in both synchronous and asynchronous FL scenarios.

Data Distributions: We consider both IID and non-IID data distributions, following the non-IID setting described in [19]. IID data simplifies training by ensuring consistent client contributions and serves as a baseline for ideal conditions. However, real-world applications, such as healthcare and mobile services, typically involve non-IID data, where heterogeneous client data can slow convergence, reduce accuracy, and amplify communication bottlenecks and dropouts.

Testing Models and Platforms: We implemented a fully synchronous FL protocol using FedAvg [19] and a fully asynchronous FL protocol using FedAsync [22], both commonly used as baselines in existing works. We selected 10 clients and varied the proportion of unreliable clients in each experiment. For model training, we used ResNet-50 [25] on the CIFAR-10 [26] dataset and another typical CNN model used as a baseline in prior studies [27] on the MNIST [28] dataset. The ResNet-50 model is a deep architecture with 50 layers and residual connections. The CNN model consists of two 5×5 convolution layers: the first layer has 20 output channels, and the second has 50, each followed by 2×2 max pooling. Evaluations were conducted on a system with an 18-core Intel® Core™ i9-7980XE CPU, 64 GB DRAM, and an NVIDIA RTX 3090 GPU.

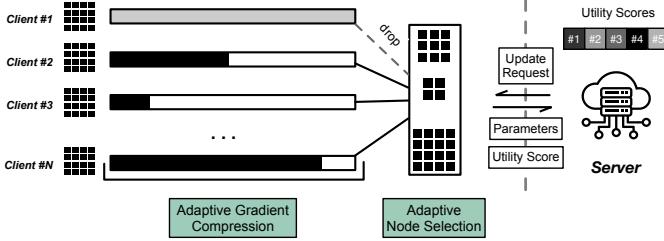


Fig. 2. Overview of AdaFL.

Results: Figure 1 shows the measurement results from our empirical study, with all experiments repeated 10 times to reduce randomness. Figure 1(a) to Figure 1(h) shows that 10% to 20% dropout and data loss had a negligible impact on overall performance in most cases. However, data loss exhibited a more pronounced negative effect due to the noise introduced by stragglers. In terms of model and dataset complexity, the impact of a higher proportion of stragglers was more significant, as shown in Figure 1(a),(e). Interestingly, in the CIFAR-10 experiments under non-IID settings, scenarios with 10% to 20% stragglers outperformed the baseline, as shown in Figure 1(c). In asynchronous settings, staleness consistently led to lower accuracy and convergence rates across all conditions than dropouts. Due to staleness, it requires a much longer time to reach a similar accuracy compared to dropouts, as shown in Figure 1(i) to Figure 1(l).

Insights: We derived two key insights:

- A small amount of client dropout and data loss have a minimal negative impact on the overall testing accuracy and convergence rate in synchronous and asynchronous FL settings. This leaves headroom for the optimization of client selection to reduce communication costs. Notably, for non-IID data distributions, the client participation rate has a limited effect on overall testing accuracy. This observation is consistent with the findings of Li et al. [29], which proves that, under non-IID settings, the convergence rate weakly replies to the client participation ratio. In a real-world deployment, client data is naturally non-IID, indicating FL robustness to partial client participation under such data distributions.
- In asynchronous FL, the effect of staleness (outdated updates) has a more significant impact on overall testing accuracy compared to client dropouts, highlighting the sensitivity of asynchronous protocols to temporal misalignment. Chen et al. [30] also demonstrate that staleness can increase the variances and affect the convergence of the global model. This motivates us to adjust gradient granularity due to network constraints adaptively.

IV. DESIGN

Overview: Our empirical study reveals that FL exhibits a degree of tolerance to lower client participation rates, with no significant negative impact on overall training performance. However, staleness has a more detrimental effect on model accuracy than client dropouts. Building on these insights,

we propose a utility and network connectivity guided FL framework, AdaFL, as shown in Figure 2. Unlike traditional optimization approaches that rely on link conditions, AdaFL leverages gradient similarity to calculate a utility score and adaptively select clients for contributing to the global model. We eliminate unnecessary communication costs associated with lower-impact contributions by prioritizing clients with higher utility scores. To further minimize communication overhead and mitigate staleness effect, AdaFL incorporates deep gradient compression [10], dynamically adjusting the compression rate based on feedback from utility score ranking. This combined approach optimizes communication efficiency while maintaining robust model performance.

Algorithm 1: Adaptive Node Selection

```

Input: Global model gradient vector  $g_G \in \mathbb{R}^d$ , Set of local gradients  $\{g_i\}_{i=1}^N$ , where  $g_i \in \mathbb{R}^d$  for client  $i$ , Number of clients to select  $K \in \mathbb{N}$ , where  $1 \leq K \leq N$ , Utility threshold  $\tau \in [0, 1]$ 
Output: Selected client set  $C_{selected} \subseteq \{1, \dots, N\}$ 
Initialize:  $C_{selected} \leftarrow \emptyset$ 
Utility Score Computation:
for  $i \leftarrow 1$  to  $N$  do
    Compute utility score:  $S_i \leftarrow f(B_i^{down}, B_i^{up}, U(g_i, \hat{g}))$ 
end
Client Filtering:
 $C_{filtered} \leftarrow \{i \in \{1, \dots, N\} : S_i \geq \tau\}$ 
Client Ranking and Selection:
Sort  $C_{filtered}$  by  $S_i$  in descending order
 $K' \leftarrow \min(K, |C_{filtered}|)$ 
 $C_{selected} \leftarrow$  First  $K'$  elements of sorted  $C_{filtered}$ 
Return:  $C_{selected}$ 
Subject to:

- $|C_{selected}| \leq K$
- $\forall i \in C_{selected} : S_i \geq \tau$
- $\forall i \in C_{selected}, j \notin C_{selected} : S_i \geq S_j$

```

Adaptive Node Selection: Our extensive experiments reveal that occasionally excluding certain clients does not negatively impact convergence accuracy, especially under non-IID settings. This observation suggests that a smaller participation ratio can be set to alleviate the straggler's effect without hurting model accuracy. Building on this insight, we propose an adaptive node selection algorithm to optimize client participation. In each training round, our approach dynamically prioritizes selecting the most impactful clients based on their contributions to the global model. Upon receiving the global model from the server, the client interrupts its current training process to calculate a utility score. This score determines whether the client should update its local model to align with the global model or continue with its existing training state, minimizing additional communication and computation overhead. The utility score S_i for each client i is defined as:

$$S_i = f(B_i^{down}, B_i^{up}, U(g_i, \hat{g})) \quad (6)$$

Here, S_i is a function of bandwidth (B_i^{down}, B_i^{up}) and similarity metric U , which measures the similarity between the gradient of the local model g_i and the gradient of the global model \hat{g} from previous round. We use cosine similarity, commonly used in existing ML works [31], [32], to measure the similarity between two models. Other metrics, such as L^2 norm and Euclidean distance [33], can also be used as a similarity metric to measure the local gradient importance toward the global

model. A high utility score between local and global gradients indicates alignment, suggesting that a client’s update will likely contribute positively to the global model’s convergence. Conversely, a low utility score represents misalignment, indicating potential noise or convergence impediments. Based on this utility score, we establish a ranking system that prioritizes clients whose gradient directions align more closely with the global gradient, as their updates are more likely to enhance model performance. Furthermore, because cosine similarity is directionally sensitive, it captures both positive and negative gradient directions. This directional awareness allows it to remain robust even during oscillations caused by high learning rates (typically in SGD), ensuring consistent detection of essential gradient patterns. To effectively manage heterogeneous data distributions across the client network and mitigate the impact of these oscillations, AdaFL integrates a warm-up stage designed to stabilize directional oscillations early in training. During these crucial early rounds, we maintain equal participation from all clients, enabling the global model to adapt gradually to diverse data patterns without premature specialization. Following this warm-up period, AdaFL transitions to a selective participation model, where only top-ranked clients—determined by the utility score—contribute to each training round. This adaptive selection strategy strikes an optimal balance between computational efficiency and model convergence quality. This adaptive selection algorithm is detailed in Algorithm 1.

Adaptive Gradient Compression: The second component addresses the critical challenge of communication overhead in FL through an adaptive compression strategy. While gradient compression effectively reduces communication costs, it must be carefully calibrated to prevent convergence degradation. We introduce a dynamic compression mechanism that changes the compression ratio based on the utility score (equation 6), preserving essential information while minimizing bandwidth consumption. Building upon deep gradient compression (DGC) [10], our approach selectively transmits significant updates while accumulating minor gradients locally. The compression ratio is adjusted based on the utility score: clients with higher utility scores receive less compression to preserve important information, while updates from less critical clients are compressed more aggressively. This selective transmission strategy reduces bandwidth requirements while maintaining model convergence through local gradient accumulation. To enhance compression effectiveness, we integrate two key DGC components: momentum correction and local gradient clipping. Momentum correction harmonizes sparse updates with dense updates temporally, mitigating convergence issues from delayed transmissions. Local gradient clipping, applied pre-accumulation, prevents gradient explosion and maintains training stability under high compression rates. During initial warm-up rounds, we maintain low compression rates across all clients to ensure robust model initialization. As the training progresses, compression rates adapt continuously based on the utility score, optimizing the balance between communication

efficiency and update significance.

TABLE I
SYNCHRONOUS FL EVALUATION RESULTS

Synchronous FL Methods	# Clients	Particip. Rate	Update Freq.	Cost Reduc. (%)	Gradient Size	Compress. Ratio	Top 1 Accuracy (IID / non-IID)
FedAvg	10	0.5	400	-50%	1.64MB	1x	MNIST: 93.06% / 86.88% CIFAR-100: 62.34% / 55.83%
FedAdam	10	0.5	400	-50%	1.64MB	1x	MNIST: 93.06% / 86.62% CIFAR-100: 61.76% / 54.92%
FedProx	10	0.5	400	-50%	1.64MB	1x	MNIST: 93.31% / 85.72% CIFAR-100: 62.02% / 54.89%
SCAFFOLD	10	0.5	400	-50%	1.64MB	1x	MNIST: 93.13% / 90.35% CIFAR-100: 61.80% / 57.73%
AdaFL	10	Adaptive	233	-70.88%	8 - 420KB	210x - 4x	MNIST: 93.43% / 87.47% CIFAR-100: 61.86% / 56.29%

TABLE II
ASYNCHRONOUS FL EVALUATION RESULTS

Asynchronous FL Methods	# Clients	Particip. Rate	Update Freq.	Cost Reduc.	Gradient Size	Compress. Ratio	Top 1 Accuracy (IID / non-IID)
FedAsync	10	0.5	400	-50%	1.64 MB	1x	MNIST: 93.25% / 85.69% CIFAR-100: 61.39% / 54.28%
FedBuff	10	0.5	400	-50%	1.64 MB	1x	MNIST: 93.72% / 88.02% CIFAR-100: 62.21% / 56.72%
AdaFL	10	Adaptive	172	-78.5%	16KB - 400KB	105x - 4x	MNIST: 93.58% / 89.11% CIFAR-100: 62.14% / 57.69%

V. EVALUATION

We evaluated AdaFL with other FL methods to answer the following three questions:

Q1. How does AdaFL perform (testing accuracy rate) compared with other optimization methods?

Q2. Can AdaFL significantly reduce communication cost while ensuring training performance?

Q3. What is the overhead of AdaFL?

Experimental Settings: We compared AdaFL to existing state-of-the-art approaches in both synchronous and asynchronous FL. In the synchronous FL domain, we benchmarked our method against FedAvg [19], FedAdam [34], FedProx [20], and SCAFFOLD [21]. Under synchronous context, AdaFL used $top - k$ topology setting, where we perform weights averaging from k clients for each communication round. For asynchronous FL comparisons, we evaluated against FedAsync [22] and FedBuff [35]. Under asynchronous context, AdaFL adapts fully asynchronous FL, where the server upgrades its global model each time it receives a gradient update. To ensure fair comparisons, we maintained consistency across all experiments using the same CNN architecture and MNIST dataset from our empirical study. We chose $r_p = 0.5$ as our participation rate out of 10 clients and strictly adhered to other hyperparameters for each baseline method while conducting all evaluations on our established experimental platform. We conduct all experiments under a fixed bandwidth to observe the reduction cost. To further validate the effectiveness of AdaFL on more complex models and datasets, we evaluated it using VGG-Net [36] on CIFAR-100 [26]. Additionally, we conducted experiments with 20 to 100 clients to assess its scalability, demonstrating its robustness across varying client participation levels.

Effectiveness: Figure 3 plots the model performance of evaluated frameworks. It can be observed that AdaFL has a higher convergence rate than existing works, as our training process is dynamically adapted and guided by the utility score.

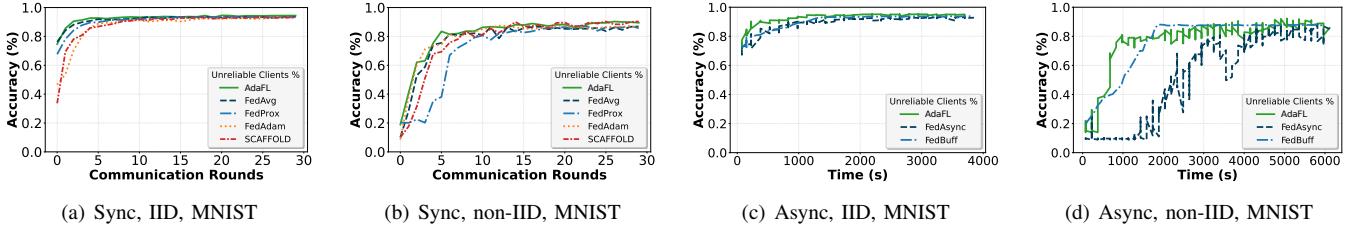


Fig. 3. Testing accuracy of CNN on MNIST for Synchronous and Asynchronous FL Protocols.

All frameworks we compared have a fixed participation rate, while ours are adjusted dynamically where $k \leq 5$. As shown in Figure 3, our learning curve is higher and more stable, especially under non-IID settings. In Figure 3(b), AdaFL has higher overall testing accuracy and a more stable learning curve from round 1 to round 5, showing the effectiveness of AdaFL under non-IID data distributions. In addition, AdaFL has a significantly higher convergence rate for asynchronous FL under non-iid settings than other frameworks, as shown in Figure 3(d). At $T = 1000s$, AdaFL reached around 80% testing accuracy where FedAsync is around 10% and FedBuff is around 50%. FedBuff has a lower noise in non-IID training due to its optimization of the buffer for weight averaging. However, AdaFL still has a higher convergence rate at the beginning of the training and can converge to a higher testing accuracy of 89.11% compared to 88.02% for FedBuff.

Answer to Q1. Under both IID and non-IID data distributions, AdaFL can outperform other methods with a higher convergence rate and testing accuracy. Specifically, under non-IID data distributions, the accuracy improvements ranged from 5% up to 70%. The results indicate the importance of the utility score guided training, especially under non-IID settings.

Communication Reduction: Table I and Table II summarizes the communication cost reduced from AdaFL. For all methods we compared, we set a fixed participation rate of $r_p = 0.5$ for each communication round, while ours has a dynamic participation rate of $r_p \leq 0.5$ where r_p is a positive value. As a result, all compared methods have a fixed update frequency, where clients update 400 times to the server for global model updates. We consider the ideal update frequency to be 800 times when all clients participate. In synchronous settings, AdaFL achieved an average of 233 update frequencies to the server, which reduced 42% unnecessary updates compared to other methods and reduced 70.88% overall communication costs. Similarly, in asynchronous settings, AdaFL had an average of 172 update frequencies, which reduced 57% of updates compared to others and reduced 78.5% overall communication costs. On the other hand, we further reduced communication costs by applying gradient compression, which AdaFL reached a compression ratio of up to 210x. This allows the server and clients to transmit gradient updates faster, which is beneficial under asynchronous contexts where the server can have a higher convergence rate from more frequent updates.

Answer to Q2. AdaFL reduces communication costs significantly in two aspects: update frequencies and gradient size.

While the communication cost is reduced, AdaFL performance is still robust, as discussed previously.

Overhead: We break down the overhead into two individual components: overhead on utility score calculation and gradient compression. We performed an ablation study using a Raspberry Pi cluster to validate our framework’s adaptability across heterogeneous devices. Our experimental setup comprised a ten-node cluster configuration, where we trained a CNN model on the MNIST dataset. We utilized *perf*, a Linux performance profiling tool, and recorded CPU cycle counts with and without AdaFL to measure the performance overhead induced by the two components. With a baseline cycle count of 8,589,175,469,216, our experimental results show a maximum of cycle counts for utility score calculation is 8,783,108,492,282, which is only around 0.05% more than the baseline. On the other hand, the performance overhead added for gradient compression is larger than the utility score calculation. However, clients can save computational resources from training and gradient compression based on the utility score. If the utility score for a client is low, the client can halt training until it can provide meaningful gradients to the server (i.e., wait for the next global update). As a result, the cost reduction from Table I and II also shows computational cost reduction. The overall overhead is negligible compared to the cost reduction benefit from the utility score calculation.

Answer to Q3. AdaFL adds additional overhead more significantly from gradient compression than utility score calculation. However, the computational cost reduction benefit from the adaptive client selection process significantly balances the overhead from gradient compression computation. Hence, the overall overhead is negligible.

VI. CONCLUSION

This paper empirically studies factors affecting FL performance, revealing resilience to client dropouts. Leveraging this, we introduce AdaFL for online adaptation. Evaluations show AdaFL outperforms SOTA FL methods by up to 30% in testing accuracy while ensuring scalability and robustness in dynamic environments. Additionally, AdaFL incurs negligible computational overhead.

ACKNOWLEDGMENT

We thank the reviewers for their valuable feedback. This work was partially supported by the NSF (CNS-2154930, CNS-2229427), ARO (W911NF-24-1-0155), and ONR (N00014-24-1-2730, N00014-24-12663).

REFERENCES

- [1] N. Wang, Y. Xiao, Y. Chen, N. Zhang, W. Lou, and Y. T. Hou, "Squeezing more utility via adaptive clipping on differentially private gradients in federated meta-learning," in *Proceedings of the 38th Annual Computer Security Applications Conference*, pp. 647–657, 2022.
- [2] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–23, 2022.
- [3] Y. Chang, H. Liu, E. Jaff, C. Lu, and N. Zhang, "Sok: Security and privacy risks of medical ai," *arXiv preprint arXiv:2409.07415*, 2024.
- [4] J. Morgan, "Federated learning meets blockchain." <https://www.jpmorgan.com/technology/news/federated-learning-meets-blockchain>, 2024. Accessed: 2024-11-18.
- [5] L. Babun, Z. B. Celik, P. McDaniel, and A. S. Uluagac, "Real-time analysis of privacy-(un) aware iot applications," *Proceedings on Privacy Enhancing Technologies*, 2021.
- [6] H. Chen, J. Ding, E. W. Tramel, S. Wu, A. K. Sahu, S. Avestimehr, and T. Zhang, "Self-aware personalized federated learning," *Advances in Neural Information Processing Systems*, vol. 35, pp. 20675–20688, 2022.
- [7] H. Liu, Y. Wu, Z. Yu, and N. Zhang, "Please tell me more: Privacy impact of explainability through the lens of membership inference attack," in *2024 IEEE Symposium on Security and Privacy (SP)*, pp. 4791–4809, IEEE, 2024.
- [8] L. Zhu, H. Lin, Y. Lu, Y. Lin, and S. Han, "Delayed gradient averaging: Tolerate the communication latency for federated learning," *Advances in Neural Information Processing Systems*, vol. 34, pp. 29995–30007, 2021.
- [9] F. Zhang, X. Liu, S. Lin, G. Wu, X. Zhou, J. Jiang, and X. Ji, "No one idles: Efficient heterogeneous federated learning with parallel edge and server computation," in *International Conference on Machine Learning*, pp. 41399–41413, PMLR, 2023.
- [10] Y. Lin, S. Han, H. Mao, Y. Wang, and W. J. Dally, "Deep gradient compression: Reducing the communication bandwidth for distributed training," *arXiv preprint arXiv:1712.01887*, 2017.
- [11] D. Alistarh, D. Grubic, J. Li, R. Tomioka, and M. Vojnovic, "Qsgd: Communication-efficient sgd via gradient quantization and encoding," *Advances in neural information processing systems*, vol. 30, 2017.
- [12] Z. Chai, Y. Chen, A. Anwar, L. Zhao, Y. Cheng, and H. Rangwala, "Feddat: A high-performance and communication-efficient federated learning system with asynchronous tiers," in *Proceedings of the international conference for high performance computing, networking, storage and analysis*, pp. 1–16, 2021.
- [13] W. Wen, C. Xu, F. Yan, C. Wu, Y. Wang, Y. Chen, and H. Li, "Terngrad: Ternary gradients to reduce communication in distributed deep learning," *Advances in neural information processing systems*, vol. 30, 2017.
- [14] J. Wangni, J. Wang, J. Liu, and T. Zhang, "Gradient sparsification for communication-efficient distributed optimization," *Advances in Neural Information Processing Systems*, vol. 31, 2018.
- [15] H. Tang, C. Yu, X. Lian, T. Zhang, and J. Liu, "Doublesqueeze: Parallel stochastic gradient descent with double-pass error-compensated compression," in *International Conference on Machine Learning*, pp. 6155–6165, PMLR, 2019.
- [16] J. Li and H. Huang, "Resolving the tug-of-war: a separation of communication and learning in federated learning," *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [17] Y. Xiong, R. Wang, M. Cheng, F. Yu, and C.-J. Hsieh, "Feddm: Iterative distribution matching for communication-efficient federated learning," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 16323–16332, 2023.
- [18] H.-P. Wang, S. Stich, Y. He, and M. Fritz, "Progfd: Effective, communication, and computation efficient federated learning by progressive training," in *International Conference on Machine Learning*, pp. 23034–23054, PMLR, 2022.
- [19] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.
- [20] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.
- [21] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "Scaffold: Stochastic controlled averaging for federated learning," in *International conference on machine learning*, pp. 5132–5143, PMLR, 2020.
- [22] C. Xie, S. Koyejo, and I. Gupta, "Asynchronous federated optimization," *arXiv preprint arXiv:1903.03934*, 2019.
- [23] Y. Fraboni, R. Vidal, L. Kameni, and M. Lorenzi, "A general theory for federated optimization with asynchronous and heterogeneous clients updates," *Journal of Machine Learning Research*, vol. 24, no. 110, pp. 1–43, 2023.
- [24] E. Ekaireb, X. Yu, K. Ergun, Q. Zhao, K. Lee, M. Huzaifa, and T. Rosing, "ns3-fl: Simulating federated learning with ns-3," in *Proceedings of the 2022 Workshop on ns-3*, pp. 97–104, 2022.
- [25] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016.
- [26] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.
- [27] H. Wang, Z. Kaplan, D. Niu, and B. Li, "Optimizing federated learning on non-iid data with reinforcement learning," in *IEEE INFOCOM 2020-IEEE conference on computer communications*, pp. 1698–1707, IEEE, 2020.
- [28] L. Deng, "The mnist database of handwritten digit images for machine learning research," *IEEE Signal Processing Magazine*, vol. 29, no. 6, pp. 141–142, 2012.
- [29] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of fedavg on non-iid data," *arXiv preprint arXiv:1907.02189*, 2019.
- [30] M. Chen, B. Mao, and T. Ma, "Efficient and robust asynchronous federated learning with stragglers," in *International Conference on Learning Representations*, 2019.
- [31] H. Qin, S. Rajbhandari, O. Ruwase, F. Yan, L. Yang, and Y. He, "Simigrad: Fine-grained adaptive batching for large scale training using gradient similarity measurement," *Advances in Neural Information Processing Systems*, vol. 34, pp. 20531–20544, 2021.
- [32] M. Fang, J. Liu, N. Z. Gong, and E. S. Bentley, "Aflguard: Byzantine-robust asynchronous federated learning," in *Proceedings of the 38th Annual Computer Security Applications Conference*, pp. 632–646, 2022.
- [33] Z. Wang, Q. Hu, X. Zou, P. Hu, and X. Cheng, "Can we trust the similarity measurement in federated learning?," *IEEE Transactions on Information Forensics and Security*, 2025.
- [34] S. Reddi, Z. Charles, M. Zaheer, Z. Garrett, K. Rush, J. Konečný, S. Kumar, and H. B. McMahan, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2020.
- [35] J. Nguyen, K. Malik, H. Zhan, A. Yousefpour, M. Rabbat, M. Malek, and D. Huba, "Federated learning with buffered asynchronous aggregation," in *International Conference on Artificial Intelligence and Statistics*, pp. 3581–3607, PMLR, 2022.
- [36] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," *arXiv preprint arXiv:1409.1556*, 2014.