EdgeLLM: Towards LLM-Guided Adaptive Distributed Training at the Edge

Reo Kuchida*
University of Tartu
Tartu, Estonia
reo.kuchida@ut.ee

Diego Salazar
Toluca Institute of Technology
Toluca, Mexico
diego.salazar@toluca.tecnm.mx

Zhigang Yin
University of Tartu
Tartu, Estonia
zhigang.yin@ut.ee

Huber Flores
University of Tartu
Tartu, Estonia
huber.flores@ut.ee

ABSTRACT

Distributed machine learning enables collaborative model training without centralizing sensitive data, yet achieving efficient and stable convergence in federated learning remains challenging due to heterogeneous data distributions, diverse device capabilities, and dynamic local operating conditions. Existing methods partially address these issues but typically rely on fixed heuristics and limited observability of system and data characteristics. In this paper, we present EdgeLLM, an LLM-guided system architecture that integrates on-device language model reasoning at the edge to infer system conditions and guide client selection and aggregation strategies for improved distributed training. Through rigorous evaluation on a real-world testbed with an NVIDIA Jetson Orin Nano edge server and 10 Raspberry Pi FL clients, EdgeLLM improves federated learning performance and robustness, achieving a weighted F1 score of 0.880 while generating structured, machine-readable reports for training adaptation. Our work paves the way for using LLMs at the edge as agents to enhance the observability and adaptability of distributed learning systems.

CCS CONCEPTS

• **Ubiquitous and pervasive computing → System modeling and simulation**.

KEYWORDS

Distributed machine learning, LLM, Federated learning

1 INTRODUCTION

Distributed Machine Learning (DML) enables collaborative AI model training while preserving data privacy by avoiding centralized data collection. Federated learning (FL) has emerged as a key paradigm in this setting, enabling privacy-preserving AI across domains such as mobile intelligence, healthcare analytics, autonomous vehicles, smart manufacturing, and Internet of Things (IoT) systems [14]. Beyond conventional FL, Federated Continuous Learning (FCL) addresses dynamic real-world environments where models must adapt to evolving data distributions while mitigating temporal catastrophic forgetting [1]. However, efficient and reliable convergence remains challenging due to heterogeneity in device capabilities, data distributions, and operating conditions. Recent advances in Large Language Models (LLMs) provide new opportunities to enhance DML by enabling more adaptive, intelligent, and autonomous optimization strategies. Realizing these benefits requires rethinking system architectures and deployment approaches to integrate LLM-based components while preserving the efficiency, scalability, and reliability required by DML applications.

Existing approaches to improve DML efficiency and robustness have explored client selection, adaptive aggregation, resource-aware scheduling, and gradient-based adaptation [2, 10, 33]. While effective, these methods are typically designed for specific scenarios and often require additional gradient analysis, clustering, or regularization, limiting their applicability in dynamic edge environments where system operators have incomplete visibility into client behavior and cannot reliably identify the specifics of each situation. Meanwhile, LLMs have demonstrated powerful capabilities for analyzing complex data and enabling adaptive decision-making [18], but cloud-based deployment introduces significant challenges, including infrastructure dependency, variable communication latency, and exposure of client sensitive data. Bridging this gap requires exploring how LLM capabilities can be distributed across the computing continuum, particularly at the edge, to enhance DML adaptability without compromising efficiency and client privacy.

In this paper, we present EdgeLLM, an innovative system architecture that augments FL with LLM capabilities at the edge. The emergence of Small Language Models (SLMs) deployable on resource-constrained devices, such as Qwen, combined with the increasing availability of GPU-accelerated edge platforms, such as NVIDIA Jetson Orin Nano, enables advanced reasoning capabilities closer to data sources (aka edge agents [16]). Thus, we introduce EdgeLLM as a complementary component that supports FL training by analyzing lightweight client metadata and providing adaptive guidance for model configuration, client selection, and aggregation strategies. When compared with LLM-based approaches, EdgeLLM preserves FL privacy properties by processing sensitive information locally, while clients share only high-level metadata related to data characteristics, training progress, and system conditions. Through rigorous validation on a real-world testbed comprising an NVIDIA Jetson Orin Nano edge server and 10 Raspberry Pi FL clients, we demonstrate that EdgeLLM can effectively fine-tune the FL process, addressing key DML challenges with a weighted F1 score of 0.880 while generating machine-readable reports. Our results show the

*Corresponding Author

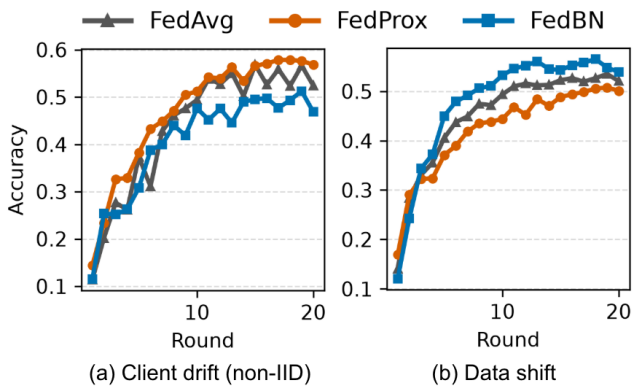

Figure 1: Model convergence behavior of different methods under different situations.

potential of SLMs as practical edge-side orchestrators for improving the observability, adaptability, and efficiency of distributed learning systems, enabling more effective and autonomous training at the edge.

2 MOTIVATION: FL LIMITATIONS

Procedure: We implement two FL degradation scenarios: client drift and data shift, on CIFAR-10 with a simple CNN using 10 clients with 5 selected per round. Client drift is induced by creating a non-IID label distribution across clients using a Dirichlet partition with $\alpha = 0.1$. Data shift is introduced through client-specific image preprocessing transformations. We use FedAvg as the baseline, FedProx [20] as a mitigation method for client drift, and FedBN [21] as a mitigation method for feature-level data shift.

Result: Figure 1 shows the test accuracy across communication rounds. FedProx achieves the best performance under client drift (58% accuracy), while FedBN performs best under data shift (57% accuracy). However, mismatched mitigation methods can even underperform FedAvg baseline, with FedBN degrading under client drift (-5%) and FedProx degrading under data shift (-3%).

Insight: Although a variety of FL techniques exist, their effectiveness is often tightly coupled to data characteristics and client participation patterns that change over time, making static strategies quickly suboptimal. This creates a gap in current FL: the lack of autonomous, real-time diagnosis of training inefficiencies under limited observability. This motivates edge-side analysis of lightweight client metadata to identify likely causes of performance degradation and communicate it to the cloud aggregation server, enabling adaptive FL strategies.

3 EDGELLM: SYSTEM DESIGN

We explore a hierarchical architecture consisting of client, edge, and cloud layers. The edge servers execute SLMs and guide the FL process by generating informative reports for the cloud aggregation server.

Design rationale: FL preserves privacy by limiting shared information. Even training-related metadata can be privacy-sensitive such as label distributions, loss histories, and device profiles. These may reveal client data characteristics or operating environments, and thus should be protected [5]. However, this limits the server's ability to exploit such information, even though it could help diagnose training dynamics and improve convergence [6]. To address this tension, we adopt an edge-centric design: edge servers are closer to clients and can process sensitive metadata within a more limited spatial and temporal scope [32]. At this layer, local SLMs generate context-aware reports by interpreting client metadata together with signals such as environmental changes, client availability, and recent training history. Unlike fixed rule-based monitoring, this approach can be flexibly adapted through prompts to better match the dynamic nature of FL.

System model: We assume that communication between clients and edge servers, as well as between edge and the cloud server, is protected by secure protocols to prevent information leakage and tampering. Under these assumptions, the additional edge layer does not fundamentally change the trust model of conventional FL, while enabling edge-based metadata processing and monitoring.

Pipeline: Figure 2 presents a high-level overview of the proposed FL architecture. In addition to the conventional FL pipeline, we introduce an edge-based monitoring process. Each client first generates a metadata report containing information such as label distribution, recent training loss, and gradient statistics, and opportunistically sends it to the nearest edge server. The edge server collects these reports and aggregates them into a summary, which is then analyzed by the local SLM to produce a training status report. When training appears stable, the report confirms the current convergence trend; when degradation is detected, it estimates which FL challenge is most likely responsible. This edge-side diagnosis is sent to the cloud aggregation server, which uses reports from multiple edge servers to obtain a higher-level view of the ongoing FL process. Based on this information, the cloud server selects an appropriate FL strategy for the current situation and continues the next round of federated training. We describe the three phases in detail below.

(i) Client metadata report creation FL clients hold useful information about their local data, system status, and training dynamics. Each client creates a metadata report containing a label histogram, device-side information such as battery level, uplink capacity, and local training time, and training-related signals including recent loss history, relative update norm, and a compressed model update direction [24]. Depending on the data modality, the report may also include simple raw-data statistics; for image datasets, per-channel RGB distributions can help detect environment-induced data drift. These signals are packed into a single JSON report and opportunistically shared with the nearest edge server.

(ii) Edge server processing The edge server collects metadata reports from multiple clients and performs preprocessing to construct a summary report. This step computes deterministic statistics such as loss trends, update norms, label imbalance, degraded labels, client availability, and system-side signals. The local SLM then takes the current summary report together with a baseline profile and diagnoses the current FL condition. The baseline profile is obtained from controlled FL runs in which the model is expected to converge under stable conditions. By comparing current training signals against this baseline, the SLM can identify whether observed values are

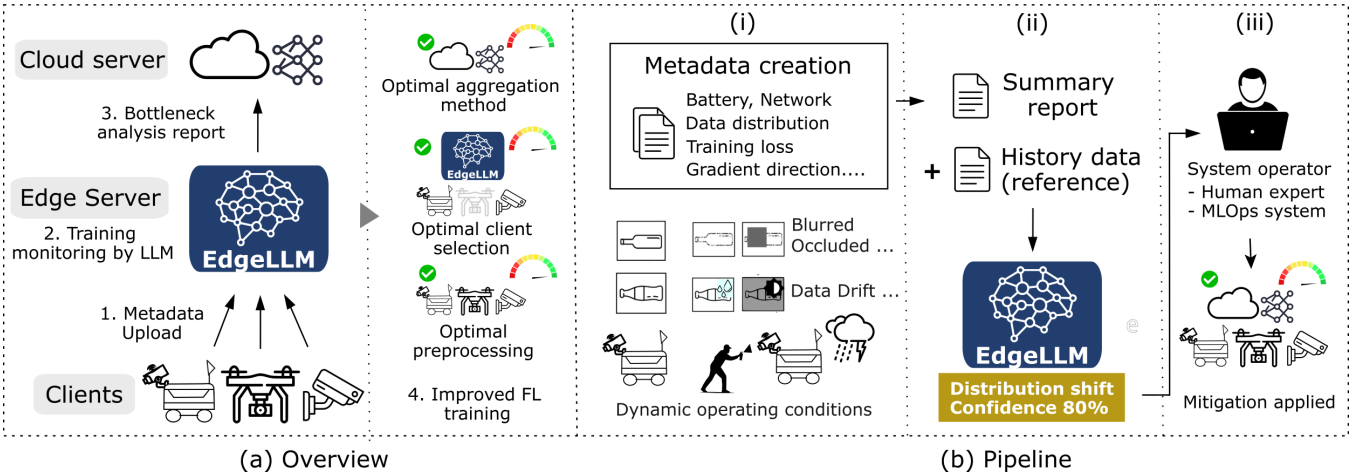

Figure 2: EdgeLLM augmentation; (a) System architecture integration: Edge on-device AI augmenting the FL architecture. (b) 3-phase pipeline.

unusually high, low, or unstable rather than interpreting them in isolation. Based on this comparison, it classifies the training state into the FL challenges discussed in Section 2. The resulting edge report, including the predicted challenge and confidence score, is then sent to the cloud aggregation server.

(iii) Cloud server adaptation Using monitoring reports from edge servers, the cloud aggregation server determines which FL strategy should be applied to the current training condition. When reports are received from multiple edge servers, a human operator or an automated MLOps system considers both the diagnosed challenges and their confidence scores in selecting suitable mechanisms, such as aggregation adjustment or client selection, for subsequent training. In principle, this adaptation can be performed at every round. In practice, however, some patterns, such as persistent loss trends or recurring client instability, require observations over multiple rounds. In this work, the cloud server analyzes the outcomes of 20 training rounds together with the accumulated edge reports to make more reliable adaptation decisions.

Table 1: Prompt template used for edge-side SLM diagnosis.

System prompt (Extracted)	
You are an edge-side diagnostic agent for federated learning. Given a JSON metadata summary, identify the primary ROOT CAUSE of degraded FL performance. Choose exactly one label: statistical heterogeneity client drift, ... Calibration: Refer to the IID random-dropout reference. Decision guide: • `system_heterogeneity_dropout_stragglers`: Selected clients fail to finish or upload updates because of device, network, or runtime limits. • `selection_aggregation_bias`: Client contribution is biased because selection or aggregation overrepresents certain available or completed clients. • `non_stationarity_distribution_shift`: A large central shift in the current or evaluation feature distribution. • `statistical_heterogeneity_client_drift`: Clients locally learn well, but their updates point in conflicting directions. • `no_major_issue`: No strong signal.	
User prompt (Extracted)	
`performance`	accuracy/loss deltas; degraded labels; ...
`participation`	dropout rate; ...
`updates`	loss reduction; direction disagreement
`config`	FedAvg; local epoch;

4 THE EXPERIMENT

Testbeds and deployment: Our testbed consists of 10 Raspberry Pi 4 Model B devices as FL clients, an NVIDIA Jetson Orin Nano as the edge server, and an HP EliteBook 840 G6 as the cloud aggregation server. For simplicity, the clients and edge server connect to the cloud server over LAN, providing stable network conditions with an average round-trip time (RTT) of 102.5ms. We implement the FL system using Flower and run Qwen3.5-4B-Thinking for diagnose and Qwen3.5-4B for reformatting as a quantized local SLM on the edge server via Ollama. For prompting, we adopt a modular design [17]. When all instructions are placed in a single prompt, the SLM tends to overthink the task and fail to produce the required fixed-format JSON output [11]. To address this, we separate diagnosis from output formatting. Table 1 shows the prompts used for the SLM and the corresponding diagnostic criteria.

FL challenge implementation: We emulate four representative FL challenges: client drift due to data heterogeneity, stragglers and dropouts due to system heterogeneity, selection and aggregation bias, and distribution shift. Client drift is induced through strongly non-IID data partitions ($\alpha = 0.05$), which amplify the divergence between local and global updates. Stragglers and dropouts are induced by assigning lower participation probabilities to selected clients, causing them to complete training less frequently. Selection bias is induced by undersampling clients that contain specific target classes, making those classes underrepresented in the aggregated update. Distribution shift is induced by applying image transformations, such as brightness changes, color shifts, and noise injection, to alter the input statistics.

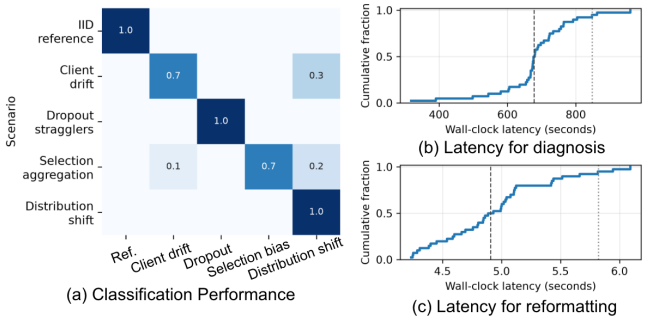

Figure 3: Classification accuracy and latency on Jetson.

Experiment procedure: We first run a standard IID FL setting as a baseline and collect client metadata reports under stable convergence. This baseline provides reference training signals for normal behavior and serves as the basis for the *no-major-issue* label. We then conduct FL experiments under the four challenge settings described above. For each setting, we run 10 independent trials, each consisting of 20 FedAvg rounds with a simple CNN. In each round, 5 clients are randomly selected, and each selected client performs 5 local epochs. After each trial, the collected client metadata are submitted to the edge-side SLM, which predicts one diagnosis label, including the four challenge types and *no-major-issue*, together with a confidence score. We evaluate the SLM in terms of classification performance (weighted F1 score) and inference time on the edge server.

5 RESULTS

SLM performance: Figure 3(a) shows the classification performance of the edge-side SLM. The SLM correctly identified the clean baseline, stragglers/dropouts, and distribution shift with 100% accuracy. For client drift caused by non-IID data and for selection/aggregation bias, the accuracy was 70%, with several cases misclassified as distribution shift. Figure 3(b) shows the inference latency on the Jetson Orin Nano. The median latency was 687 seconds for the diagnosis step and 4.7 seconds for the reformatting step, resulting in a total median latency of approximately 700 seconds. The 90th-percentile total latency was 817 seconds. Although the overhead is non-negligible compared with the average client local training time of 300 seconds in this setting, its relative impact is expected to decrease as client-side workloads become more computationally intensive. As a baseline comparison, we examined whether the cloud server could diagnose training conditions using only global aggregation-level metrics. In the clean setting, the global model achieved approximately 50% test accuracy, whereas under the challenge settings the accuracy dropped to 14–27%, with similar degradation in loss. These indicate that training performance has deteriorated, but they cannot distinguish whether the cause is client drift, stragglers, selection bias, or distribution shift.

FL improvements: Table 2 summarizes the improvements obtained using condition-specific mitigation methods. FedProx improved validation accuracy under client drift by 1.8 percentage points, while completion-aware over-selection improved performance under client dropout by 4.4 points. Label-coverage-aware selection provided a 1.6-point improvement under selection and aggregation bias. Clean-data replay achieved the largest improvement under distribution shift, increasing overall accuracy by 9.0 points. In particular, it recovered clean-domain accuracy from 0.09 to 0.19, mitigating catastrophic forgetting, while decreasing accuracy on the shifted domain by only 0.2 points.

Table 2: Mitigation methods improve the performance over the corresponding baselines.

Scenario	Mitigation method	Improvement
Client drift	FedProx	+1.8 %
Dropout	Completion-aware over-selection	+4.4 %
Selection bias	Label-coverage-aware client selection	+1.6 %
Distribution shift	Clean-data replay (20%)	+9.0 %

Reliability of SLM-based diagnosis: The SLM misclassifications were not random. Most error cases contained overlapping signals from multiple FL challenges. For example, some distribution-shift trials also showed disagreement in model update directions, which is a typical sign of client drift. When we grouped the four challenges into three broader categories, namely model-update, availability, and continuous-performance issues, the SLM correctly classified all cases. This indicates that even when fine-grained labels are ambiguous, the SLM still provides useful diagnostic information for cloud-side adaptation. The modular prompt design also improved output reliability. With modular prompting, all SLM outputs followed the instructed fixed JSON format. In contrast, in the single-prompt ablation, only 20% of the outputs satisfied the required JSON format. This result highlights that prompt decomposition is important not only for diagnosis quality but also for producing machine-readable reports that can be directly used by the cloud server.

GPU-accelerated inference: GPU acceleration is important for making edge-side SLM inference practical. The Jetson Orin Nano has a small form factor comparable to Raspberry Pi-class devices, while providing GPU acceleration for local model inference. In a controlled experiment with fixed input and output lengths of 128 tokens each, we compared the inference speed of Qwen3.5-4B on a Jetson Orin Nano GPU, a laptop CPU equipped with an Intel Core i5 processor, and a Raspberry Pi 4 CPU. As shown in Table 3, the Jetson achieved 11.77 tokens/s for decoding, which is 2.52× faster than the laptop CPU and 15.49× faster than the Raspberry Pi 4. It also achieved 112.39 tokens/s for prompt processing, corresponding to 1.69× and 10.38× speedups over the laptop CPU and Raspberry Pi 4, respectively. These are yielding 2.33× and 14.22× end-to-end speedups. These results show that GPU acceleration improves both prompt encoding and response decoding. This is especially useful for our modular prompting design, where the output of one SLM step is used as the input to the next step. On CPU-only devices, repeatedly reading and processing previous SLM outputs becomes a significant bottleneck, whereas the Jetson GPU can execute this pipeline more efficiently.

Table 3: Inference speed comparison for Qwen3.5-4B.

Device	Decode speed (tokens/s)	Prompt speed (tokens/s)	Wall time (s)
Jetson GPU	11.77	112.39	12.18
Intel i5-10th Gen CPU	4.68	66.64	28.39
RPI4 CPU	0.76	10.83	173.16
Jetson v.s. laptop	2.52×	1.69×	2.33×
Jetson v.s. RPi	15.49×	10.38×	14.22×

6 DISCUSSION

Room for improvement: EdgeLLM can be extended in two directions. First, its diagnostic performance could be improved by incorporating richer contextual sensing, including mobility, network conditions, charging state, and temporal user behavior [28]. Such information would enable more fine-grained diagnosis of FL challenges by helping identify more specific underlying causes. Second, its security could be strengthened by deploying trusted execution environments, with OS integrity protected through mechanisms such as Secure Boot, trust hardware roots, and sandboxing [4, 26].

SLM challenges and opportunities: Prompt design is crucial for reliable SLM-based reasoning at the edge. Overly specific prompts may cause the model to overfit to rigid threshold-based rules, while ambiguous prompts can lead to excessive reasoning or unnecessarily long generations. Therefore, prompts should bound the reasoning process while remaining flexible to diverse client-side conditions. Also since SLM-generated reports are consumed by downstream components, output consistency is essential. A modular prompt design is one of the solutions for organizing diagnosis, explanation, and recommendation into structured fields, which improves controllability and facilitates integration with the FL pipeline.

EdgeLLM generalization: We demonstrated that EdgeLLM can dynamically optimize the FL process by selecting the most suitable client data for training. This adaptive capability is valuable for tailoring the FL process to specific deployment scenarios. Beyond client selection, EdgeLLM could also identify opportunities for additional optimizations, such as recommending appropriate data preprocessing or augmentation strategies to improve training convergence. More broadly, EdgeLLM has the potential to diagnose the overall FL process and provide actionable recommendations that generalize across different models, datasets, and training configurations.

7 RELATED WORK

FL optimization strategies: Model convergence in FL is affected by multiple sources of heterogeneity, including non-IID client data, systems heterogeneity such as stragglers and dropouts, biased participation, distribution shift, and temporal forgetting [1, 7, 15, 19]. Prior work has proposed different solutions for these issues, such as FedProx and SCAFFOLD for client drift, resource-aware client selection for systems heterogeneity, bias-aware objectives for client distribution imbalance, and continual-learning methods for evolving data [1, 15, 20, 25]. However, these methods target different causes of degradation, and their effectiveness depends on whether the selected mitigation matches the actual training condition. *Unlike others, EdgeLLM improves FL training process through LLM guidance at the edge.*

Runtime diagnosis and adaptive mitigation in FL: Since effective mitigation depends on correctly identifying the cause of degradation, prior work has used runtime signals to detect adverse FL conditions and adapt training accordingly. Client-side confidence and loss statistics support concept-drift detection, dynamic model clustering, and utility-aware sample selection [13, 31]. System telemetry, including training latency and execution speed, further supports heterogeneity-aware optimal client selection and scheduling [3]. However, these signals are not unique to a single cause: increased loss may result from concept drift, noisy data, or unstable optimization [27]. A richer alternative is to analyze model updates using dimensionality reduction and clustering to identify drifted or anomalous clients [23, 35]. Nevertheless, such analyses remain problem-specific; deploying multiple detectors introduces substantial overhead and ambiguity when the underlying cause is unknown or multiple conditions coexist [34]. *EdgeLLM instead jointly analyzes privacy-sensitive client metadata at the edge, enabling more informed diagnosis and condition-specific mitigation.*

Edge-centric architectures in FL: Hierarchical FL, which introduces an edge layer between clients and the server, has been proposed to reduce communication overhead and improve adaptability to heterogeneous clients with diverse data distributions [22, 29]. Such architectures can mitigate non-IID convergence challenges when geographically close clients exhibit similar data distributions [8], and communication cost is optimized by increasing communication between client and edge and reducing the edge and cloud [9]. Agent-like capabilities have been proposed also in the edge layer to foster sensor data governance in IoT environments [16]. In addition, edge servers can serve as strategic defense boundaries for implementing hierarchical differential privacy, shielding sensitive local gradients before they are uploaded to an untrusted cloud server [30]. Recently, an FL framework has been proposed that deploys LLM agents at both the clients and the server [12]. *Unlike others, EdgeLLM uses LLMs at the edge to improve the FL training.*

8 SUMMARY AND CONCLUSION

We presented EdgeLLM, an LLM-guided architecture that leverages on-device language model reasoning to infer system conditions and guide client selection and aggregation strategies for improved distributed training. Rigorous evaluation on a FL testbed comprising 10 Raspberry Pi clients and an NVIDIA Jetson Orin Nano edge server equipped with a Qwen small language model shows that edge-deployed SLMs can improve FL convergence. While misclassifications are still observed, they suggest that performance could be further improved through more structured guidance of the FL process; however, this would require capturing additional information about FL dynamics during training, potentially impacting privacy guarantees. Overall, our results highlight the potential of resource-constrained language models as practical edge-side orchestrators for enhancing the adaptability and efficiency of distributed learning systems.

ACKNOWLEDGMENTS

This work was supported by Estonian Research Council grant (PRG 2725). We thank the Delfin Program for supporting the research visit of Diego Salazar. Special thanks to the reviewers for the valuable suggestions.

REFERENCES

[1] Yavuz Faruk Bakman, Duygu Nur Yaldiz, Yahya H Ezzeldin, and Salman Avestimehr. 2023. Federated orthogonal training: Mitigating global catastrophic forgetting in continual federated learning. *arXiv preprint arXiv:2309.01289* (2023).

[2] Fabiola Espinoza Castellon, Aurélien Mayoue, Jacques-Henri Sublemontier, and Cédric Gouy-Pailler. 2022. Federated learning with incremental clustering for heterogeneous data. In *2022 International Joint Conference on Neural Networks (IJCNN)*. IEEE, 1–8.

[3] Zheng Chai, Ahsan Ali, Syed Zawad, Stacey Truex, Ali Anwar, Nathalie Baracaldo, Yi Zhou, Heiko Ludwig, Feng Yan, and Yue Cheng. 2020. Tifl: A tier-based federated learning system. In *Proceedings of the 29th international symposium on high-performance parallel and distributed computing*. 125–136.

[4] Wenyu Dong, Guangquan Xu, Shicheng Feng, Hongpeng Bai, Michael Sheng, and Xi Zheng. 2025. Escorting the Confidentiality and Integrity of UAVs: What Exactly Can Trusted Execution Environment Offer? *Comput. Surveys* 58, 3 (2025), 1–35.

[5] Aidmar Wainakh et al. 2021. User-level label leakage from gradients in federated learning. *arXiv preprint arXiv:2105.09369* (2021).

[6] Jie Zhang et al. 2022. Federated learning with label distribution skew via logits calibration. In *International Conference on Machine Learning*. PMLR, 26311–26329.

[7] Keith Bonawitz et al. 2019. Towards federated learning at scale: System design. *Proceedings of machine learning and systems* 1 (2019), 374–388.

[8] Seunghyun Lee et al. 2026. Personalizing federated learning for hierarchical edge networks with non-IID data. *IEEE Internet of Things Journal* (2026).

[9] Xiaofan Yu et al. 2023. Async-HFL: Efficient and robust asynchronous federated learning in hierarchical IoT networks. In *Proceedings of the 8th ACM/IEEE Conference on Internet of Things Design and Implementation*. 236–248.

[10] Zheng Wang et al. 2023. Fedgs: Federated graph-based sampling with arbitrary client availability. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 37. 10271–10278.

[11] Zihao Wei et al. 2025. Stop spinning wheels: Mitigating llm overthinking via mining patterns for early reasoning exit. *arXiv preprint arXiv:2508.17627* (2025).

[12] Rafael O. Jarczewski, Gabriel U. Talasso, Leandro A. Villas, and Allan Mariano de Souza. 2026. Agentic Federated Learning: The Future of Distributed Training Orchestration.

[13] Ellango Jothimurugesan, Kevin Hsieh, Jianyu Wang, Gauri Joshi, and Phillip B Gibbons. 2023. Federated learning under distributed concept drift. In *International Conference on Artificial Intelligence and Statistics*. PMLR, 5834–5853.

[14] Peter Kairouz, H. Brendan McMahan, Brendan Avent, et al. 2021. Advances and Open Problems in Federated Learning. *Foundations and Trends in Machine Learning* 14, 1–2 (2021), 1–210.

[15] Sai Praneeth Karimireddy, Satyen Kale, Mehryar Mohri, Sashank Reddi, Sebastian Stich, and Ananda Theertha Suresh. 2020. Scaffold: Stochastic controlled averaging for federated learning. In *International conference on machine learning*. PMLR, 5132–5143.

[16] Musfira Khan, Fatemeh Sarhaddi, Agustin Zuniga, Huber Flores, Sasu Tarkoma, and Petteri Nurmi. 2026. Governance at the edge: Agent-driven privacy mediation for mobile and iot data. In *Proceedings of the 27th International Workshop on Mobile Computing Systems and Applications*. 85–90.

[17] Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. [n. d.]. Decomposed Prompting: A Modular Approach for Solving Complex Tasks. In *The Eleventh International Conference on Learning Representations*.

[18] Reo Kuchida, Mayowa Olapade, Kevin Post, Ashfaq Hussain Ahmed, and Huber Flores. 2025. Assembling Edge Intelligence and Decentralized Infrastructure on Command using LLMs. In *IEEE INFOCOM 2025-IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS)*. IEEE, 1–6.

[19] Fan Lai, Xiangfeng Zhu, Harsha V Madhyastha, and Mosharaf Chowdhury. 2021. Oort: Efficient federated learning via guided participant selection. In *15th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 21)*. 19–35.

[20] 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.

[21] Xiaoxiao Li, Meirui Jiang, Xiaofei Zhang, Michael Kamp, and Qi Dou. 2021. Fedbn: Federated learning on non-iid features via local batch normalization. *arXiv preprint arXiv:2102.07623* (2021).

[22] Lumin Liu, Jun Zhang, SH Song, and Khaled B Letaief. 2020. Client-edge-cloud hierarchical federated learning. In *ICC 2020-2020 IEEE international conference on communications (ICC)*. IEEE, 1–6.

[23] Dimitrios Michael Manias, Ibrahim Shaer, Li Yang, and Abdallah Shami. 2021. Concept drift detection in federated networked systems. In *2021 IEEE global communications conference (GLOBECOM)*. IEEE, 1–6.

[24] 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.

[25] Mehryar Mohri, Gary Sivek, and Ananda Theertha Suresh. 2019. Agnostic federated learning. In *International conference on machine learning*. PMLR, 4615–4625.

[26] Abdul-Rasheed Ottun, Rasinthe Marasinghe, Toluwani Elemosho, Mohan Liyanage, Mohamad Ragab, Prachi Bagave, Marcus Westberg, Mehrdad Asadi, Michell Boerger, Chamara Sandeepa, et al. 2024. The SPATIAL architecture: Design and development experiences from gauging and monitoring the AI inference capabilities of modern applications. In *2024 IEEE 44th International Conference on Distributed Computing Systems (ICDCS)*. IEEE, 947–959.

[27] Lorena Poenaru-Olaru, Luis Cruz, Arie Van Deursen, and Jan S Rellermeyer. 2022. Are concept drift detectors reliable alarming systems?-a comparative study. In *2022 IEEE International Conference on Big Data (Big Data)*. IEEE, 3364–3373.

[28] Kevin Post, Reo Kuchida, Mayowa Olapade, Zhigang Yin, Petteri Nurmi, and Huber Flores. 2025. Contextllm: Meaningful context reasoning from multi-sensor and multi-device data using llms. In *Proceedings of the 26th International Workshop on Mobile Computing Systems and Applications*. 13–18.

[29] Syeda Sanjana Shahid, Tara Salman, Mohamed Baza, and Marcio Andrey Teixeira. 2025. Hierarchical Federated Learning: Approaches, Applications, and Open Challenges. *IEEE Access* 13 (2025), 190333–190353.

[30] Lu Shi, Jiangang Shu, Weizhe Zhang, and Yang Liu. 2021. HFL-DP: Hierarchical federated learning with differential privacy. In *2021 IEEE Global Communications Conference (GLOBECOM)*. IEEE, 1–7.

[31] Jaemin Shin, Yuanchun Li, Yunxin Liu, and Sung-Ju Lee. 2022. Fedbalancer: Data and pace control for efficient federated learning on heterogeneous clients. In *Proceedings of the 20th Annual International Conference on Mobile Systems, Applications and Services*. 436–449.

[32] Valeriu-Daniel Stanciu, Maarten van Steen, Ciprian Dobre, and Andreas Peter. 2021. Privacy-preserving crowd-monitoring using bloom filters and homomorphic encryption. In *Proceedings of the 4th International Workshop on Edge Systems, Analytics and Networking*. 37–42.

[33] Guanghui Tong, Gaolei Li, Jun Wu, and Jianhua Li. 2021. Gradmfl: Gradient memory-based federated learning for hierarchical knowledge transferring over non-iid data. In *International Conference on Algorithms and Architectures for Parallel Processing*. Springer, 612–626.

[34] Wei Wan, Shengshan Hu, Minghui Li, Jianrong Lu, Longling Zhang, Leo Yu Zhang, and Hai Jin. 2023. A four-pronged defense against byzantine attacks in federated learning. In *Proceedings of the 31st ACM international conference on multimedia*. 7394–7402.

[35] Tianye Zhang, Haozhe Feng, Wenqi Huang, Lingyu Liang, Huanming Zhang, Zexian Chen, Anthony KH Tung, and Wei Chen. 2025. FedCare: towards interactive diagnosis of federated learning systems. *Frontiers of Computer Science* 19, 7 (2025), 197335.