Taming Volatility: Stable and Private QUIC Classification with Federated Learning

Richard Jozsa*, Karel Hynek†¶, and Adrian Pekar*‡§
*Budapest University of Technology and Economics, Műegyetem rkp. 3., H-1111 Budapest, Hungary
†Czech Technical University in Prague, Faculty of Information Technology, Czech Republic
‡HUN-REN-BME Information Systems Research Group, Magyar Tudósok krt. 2, 1117 Budapest, Hungary
§CUJO LLC, Budapest, Hungary
¶CESNET, a.l.e., Prague, Czech Republic
Email: richard.jozsa@edu.bme.hu, hynekkar@fit.cvut.cz, apekar@hit.bme.hu

***Abstract*—Federated Learning (FL) is a promising approach for privacy-preserving network traffic analysis, but its practical deployment is challenged by the non-IID nature of real-world data. While prior work has addressed statistical heterogeneity, the impact of temporal traffic volatility—the natural daily ebb and flow of network activity—on model stability remains largely unexplored. This volatility can lead to inconsistent data availability at clients, destabilizing the entire training process. In this paper, we systematically address the problem of temporal volatility in federated QUIC classification. We first demonstrate the instability of standard FL in this dynamic setting. We then propose and evaluate a client-side data buffer as a practical mechanism to ensure stable and consistent local training, decoupling it from real-time traffic fluctuations. Using the real-world CESNET-QUIC22 dataset partitioned into 14 autonomous clients, we then demonstrate that this approach enables robust convergence. Our results show that a stable federated system achieves a 95.2% F1 score, a mere 2.3 percentage points below a non-private centralized model. This work establishes a blueprint for building operationally stable FL systems for network management, proving that the challenges of dynamic network environments can be overcome with targeted architectural choices.**

***Index Terms*—QUIC protocol, federated learning, traffic classification, network security, privacy preservation, neural networks**

I. Introduction

Network traffic classification (TC) has become increasingly challenging with the widespread adoption of encrypted protocols, particularly QUIC, which now carries a significant portion of internet traffic [1]. The QUIC protocol's encrypted nature and connection multiplexing capabilities fundamentally alter traditional traffic analysis approaches, requiring new classification methodologies that operate on statistical flow features rather than packet payload inspection. This shift has significant implications for network operators who depend on accurate TC for capacity planning, quality of service provisioning, and security monitoring.

Current state-of-the-art QUIC classification approaches rely heavily on centralized machine learning models trained on aggregated network flow data. While recent work has demonstrated that neural networks and ensemble methods can achieve high classification accuracy on QUIC traffic [2], [3], these approaches assume the availability of comprehensive, centralized datasets spanning multiple network environments. This assumption becomes problematic in real-world deployments where network operators face strict privacy regulations, competitive concerns, and data sovereignty requirements that prohibit sharing raw traffic data across organizational boundaries.

The fundamental tension between the need for comprehensive training data and privacy preservation has led to the exploration of Federated Learning (FL) [4]. Recent work like FedETC [5] has already shown that FL is a viable privacy-preserving solution for general encrypted TC. However, beyond the initial challenge of privacy, a critical *operational challenge* remains that hinders the practical deployment of FL in real network environments: *temporal traffic volatility*.

Real-world network traffic is not static; it exhibits significant diurnal (daily) fluctuations. This temporal volatility means that clients may have insufficient data during low-traffic periods, leading to noisy model updates that can destabilize the entire synchronous training process. This operational instability, which has been largely overlooked in prior work focused on statistical non-IID data, represents a major barrier to deploying reliable and consistently high-performing FL systems for network management.

This paper is the first to systematically identify and address this stability problem in the context of multi-class QUIC service classification. We propose the novel application of a client-side buffering mechanism—a technique adapted from other FL domains where it served different purposes—to decouple local training from real-time traffic fluctuations and ensure robust model convergence. Our contributions are:

- We systematically analyze *temporal traffic volatility* as a critical destabilizing factor for synchronous FL in TC.
- We propose and evaluate the application of a *client-side buffering mechanism* as a practical and effective solution to ensure stable model convergence in dynamic network environments.
- We provide a comprehensive evaluation of FL specifically for *multi-class QUIC service classification*, using a realistic 14-client setup and real-network data.
- We demonstrate that our stable federated approach achieves near-optimal performance (95.2% F1-score), quantifying the small performance trade-off required for a private and operationally robust system.

978-3-903176-75-1 ©2025 IFIP

Our experimental evaluation uses the CESNET-QUIC22 dataset [6], partitioned to simulate 14 autonomous clients over a two-week period. Our findings demonstrate that FL, when architected with mechanisms like client-side buffering to handle the dynamics of network environments, represents a viable path toward privacy-preserving and *operationally stable*, large-scale QUIC TC.

II. Related Work

Our research is positioned at the intersection of three key domains: the classification of QUIC traffic, the application of FL to encrypted network data, and the use of stability mechanisms within FL.

A. QUIC Traffic Classification Approaches

Early research into classifying the then-new QUIC protocol focused on adapting machine learning techniques to its encrypted nature. Tong *et al.* [2] pioneered the use of convolutional neural networks (CNNs), demonstrating that statistical flow features could effectively distinguish QUIC services despite payload encryption. Building on this, Luxemburk *et al.* [3] conducted a comprehensive comparison of different models using the CESNET-QUIC22 dataset [6], confirming that aggregated flow statistics are highly discriminative. The availability of this large-scale, real-world dataset has been crucial for advancing the field. However, these state-of-the-art approaches all presuppose a centralized data collection model, which presents the significant privacy and logistical challenges that motivate decentralized paradigms like FL.

B. Federated Learning for Encrypted Traffic

The application of FL to network traffic analysis is an active area of research, primarily driven by privacy needs. Much of the existing work has focused on overcoming challenges related to statistical non-IID data. For instance, Guo *et al.* [7] and Guo and Wang [8] proposed sophisticated client selection strategies to improve convergence when data distributions differ across clients. Similarly, Pekar *et al.* [9] explored model performance under extreme non-IID conditions.

Very recently, Jin *et al.* [5] proposed FedETC, a federated framework for general encrypted TC. Their work successfully demonstrates that high-performance, privacy-preserving classification is feasible, achieving accuracy close to that of centralized models. However, these existing studies do not address the operational challenge of model stability in the face of real-world traffic fluctuations over time. This temporal volatility is particularly pronounced for QUIC, as many of its dominant applications (e.g., video streaming, social media, real-time communication) are highly interactive and user-driven, leading to pronounced and predictable daily usage cycles compared to background or automated traffic.

C. Buffering Mechanisms in Federated Learning

The concept of buffering client data or updates is not new to the FL literature, but its application has been targeted at solving system-level and scheduling inefficiencies. For example, Nguyen *et al.* [10] introduced FedBuff, a system that uses buffered asynchronous aggregation to improve scalability and tolerate "straggler" clients that are slow to respond in large-scale deployments. Other works have used buffers for more efficient client selection or to manage on-device resources. These mechanisms are designed to optimize the FL system's internal operations. To our knowledge, client-side data buffering has not been proposed or evaluated as a mechanism to specifically counteract the destabilizing effect of *temporal data volatility* originating from an external, dynamic process like network traffic.

D. Positioning and Research Gap

Our work addresses a critical, unaddressed gap at the confluence of these fields. While FL has been applied to encrypted traffic, and buffering has been used for system efficiency, the specific problem of maintaining *stable federated training* for QUIC classification in a dynamic network environment with fluctuating traffic volumes has not been solved. The primary novelty of our paper is therefore not the application of FL to private TC in general, but rather the *identification of temporal volatility as a key operational barrier* and *the novel application of a buffering mechanism to solve it*, enabling robust and practical deployment.

III. Methodology

Our experimental approach evaluates FL for QUIC TC through a simulation designed to reflect realistic network operator scenarios. We structure our methodology around three key components: a realistic data distribution that captures both organizational and temporal heterogeneity, a neural network architecture optimized for flow-level features, and a comprehensive FL framework incorporating our proposed stability mechanism.

A. Dataset and Experimental Design

We base our evaluation on the CESNET-QUIC22 dataset [6]. It contains labeled QUIC flows collected during four consecutive weeks (Weeks 44–47) from the backbone network infrastructure used by half a million users daily. Since the dataset is anonymized, we contacted the CESNET-QUIC22 authors, who partitioned the data into 14 distinct subsets. Each subset represents traffic from a different IP range associated with a specific, but unknown to us, organization inside the CESNET network.

Due to the substantial data drift observed in the first two weeks of CESNET-QUIC dataset [6], which could severely affect the experimental results, we limited our analysis to a continuous span of two weeks (Week 46 and Week 47), during which the drift appears to be present but less severe. We further restricted our study to the seven most common and diverse services—Discord, Facebook-Graph, Google-WWW, Instagram, Snapchat, Spotify, and YouTube—to maintain a focused and representative evaluation.

Each of the 14 clients trains on the data originating from their unique network prefixes within these intervals, naturally exposing the federated system to both statistical heterogeneity

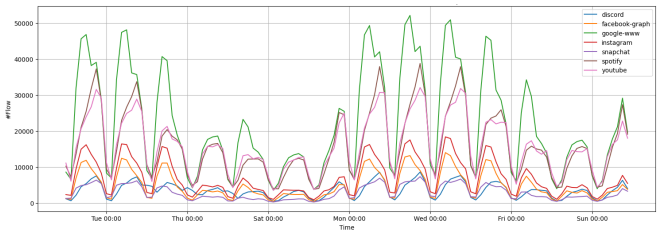


Fig. 1: Temporal distribution of services in CESNET-QUIC22 dataset over the selected two-week experimental period.

(different service usage across organizations) and temporal heterogeneity (daily traffic fluctuations).

The distribution of flows across our 3-hour intervals, illustrated in Fig. 1, reveals pronounced daily cycles common to real networks. This setup provides a powerful benchmark for evaluating the robustness of FL algorithms under practical, challenging conditions.

B. Feature Engineering

The raw CESNET-QUIC22 dataset provides basic flow statistics and packet-level information (PPI) for the first 30 packets. To create a more discriminative feature set for classification, we perform two key feature engineering steps:

- *Per-Direction Packet Features*: We disaggregate the initial packet information (size and inter-arrival time) into separate features for each communication direction (client-to-server and server-to-client). This captures the asymmetric nature of client-server interactions.
- *Per-Direction Flow Statistics*: From these directional packet features, we compute higher-level statistics, such as the mean, standard deviation, min, and max of packet sizes and inter-arrival times for each direction.

This process results in a comprehensive feature vector of $N = 283$ features for each flow, combining the original dataset's statistics with our engineered directional features. All features are scaled to a [0, 1] range using min-max scaling before being fed into the neural network.

C. Neural Network Architecture

Our classification model employs a fully connected neural network (FCN) designed for tabular flow statistics. The network uses an expanding structure to learn progressively complex feature interactions:

- *Input Layer*: N neurons (number of input features)
- *Hidden Layer 1*: $2N$ neurons + BatchNorm + LeakyReLU(0.1) + 15% Dropout
- *Hidden Layer 2*: $3N$ neurons + BatchNorm + LeakyReLU(0.1) + 15% Dropout
- *Hidden Layer 3*: $3N$ neurons + BatchNorm + LeakyReLU(0.1) + 15% Dropout
- *Hidden Layer 4*: $4N$ neurons + BatchNorm + LeakyReLU(0.1) + 15% Dropout
- *Output Layer*: 7 neurons (one for each service class)

Hidden layers incorporate batch normalization for training stability, LeakyReLU activation to prevent the dying ReLU problem, and 15% dropout for regularization. The model is trained using cross-entropy loss and the Adam optimizer with a learning rate of 0.001 and batch size of 64 for federated clients (compared to 0.01 and 1024 for centralized training) to ensure more robust gradient computation with limited local data. Each local training round consists of 10 epochs.

D. Federated Learning Configuration

Our FL simulation utilizes the Flower framework [11] with PyTorch. The protocol follows standard FL communication rounds where the server broadcasts the global model, clients train locally, and return updated parameters. We systematically evaluate five aggregation algorithms to understand their behavior in this context:

- *FedAvg* [4]: Baseline weighted parameter averaging.
- *FedProx* [12]: Adds a proximal term to handle client drift.
- *FedAdam*, *FedAdagrad*, *FedYogi* [13]: Implement server-side adaptive optimization.

E. Addressing Temporal Volatility with Client-Side Buffering

A key challenge we identified in applying synchronous FL to network traffic is the model instability caused by temporal data volatility. To address this, we propose a novel application of a *client-side FIFO (First-In, First-Out) data buffer*. In our framework, instead of training on all data collected in a 3-hour interval, each client accumulates incoming flow records into a fixed-size buffer. A local training round is only initiated once the buffer contains a sufficient number of samples (6400 records in our experiments). If new data causes the buffer to exceed its capacity, the oldest records are discarded.

This mechanism serves to decouple local model training from real-time traffic volume. By ensuring that each local update is derived from a consistent and statistically significant amount of data, the buffer smooths out the impact of temporal fluctuations, leading to more stable and robust global model convergence. This directly addresses the temporal volatility challenge identified as a key barrier to practical deployment.

F. Client-Side Data Handling

To evaluate the impact of temporal volatility, we define two distinct client-side data handling strategies:

- *Standard FL (unbuffered):* In each 3-hour round, clients use all data that arrived in that interval. This data is split into training (70%), validation (10%), and test (20%) sets. This results in highly variable dataset sizes per round, directly reflecting traffic volatility.
- *Buffered FL (our proposal):* Clients maintain three fixed-size FIFO buffers for training (6400 flows), validation (914), and testing (1828), reflecting the selected 70/10/20 ratio. In each FL round, newly arriving flows are partitioned according to this ratio and added to the respective FIFO buffers. This approach ensures stable and consistent dataset sizes for training and evaluation across rounds.

In both scenarios, the validation set is used for local early stopping, and the test set is used to report the performance for each round.

G. Evaluation Framework

Our framework assesses both classification performance and FL dynamics. We first establish a *centralized baseline* using the complete, aggregated dataset to define an upper performance bound.

We use SHAP (SHapley Additive exPlanations) [14] on the centralized model to identify the most predictive flow features under ideal conditions, providing a ground truth for interpreting the model's behavior.

Finally, for the federated scenarios, we directly compare the convergence stability of the *standard (unbuffered) FL* against our *proposed buffered FL* to quantify the impact of our mechanism. The primary performance metric reported in this paper is the macro F1 score; however, additional metrics are provided in the digital artifact [15].

IV. Experimental Results

A. Centralized Performance and Feature Insights

We first establish an upper-bound performance by training our FCN on the complete, aggregated two-week dataset. This *centralized baseline* achieves a 97.5% F1 score. The classifier showed strong per-class performance, where all classes were recognized with more than 95% precision, confirming the general separability of the QUIC services with our engineered features. Discord and Spotify are classified with over 99% precision, indicating highly distinctive traffic patterns. Social media services like Instagram (96.74%) and Facebook-Graph (95.26%) show more confusion, likely due to overlapping backend APIs and content delivery networks.

To understand which features are fundamentally most important, we performed a SHAP analysis on this centralized model. The results in Fig. 2 reveal that packet size statistics from the server dominate the rankings. Specifically, 'DST_PS_2' is the most influential feature. This aligns with domain knowledge, as this packet often contains the server's TLS certificate, whose size is a highly discriminative service signature. This analysis provides a ground truth for what an ideal model should learn.

B. Instability of Standard FL

Next, we evaluate a standard synchronous FL setup where clients train only on the data available in each 3-hour round. Fig. 3a shows the result: the system is plagued by instability. The performance of all five aggregation algorithms plummets during low-traffic nighttime periods (shaded regions), with some clients (particularly clients 10 and 14 in our experiments) failing to converge entirely due to insufficient data for reliable gradient estimation. The dashed red line represents the centralized baseline F1 score. It is clear that without a stability mechanism, standard FL consistently and significantly underperforms.

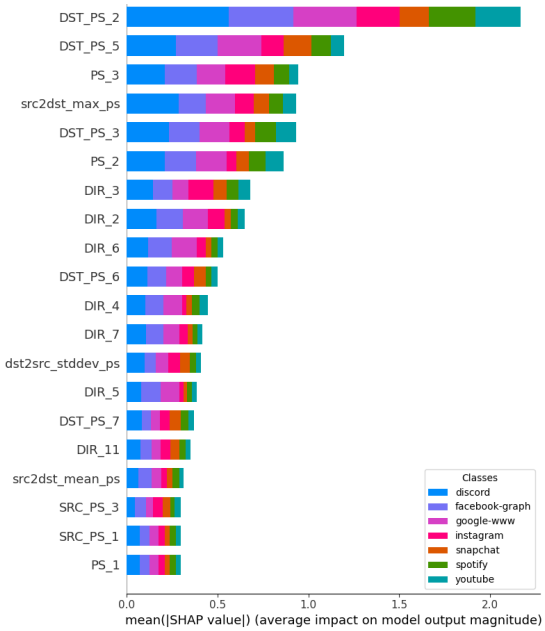


Fig. 2: SHAP analysis on the centralized model identifies the most influential flow features for QUIC classification. `PS_n` denotes the size of the *n*-th packet in the bidirectional flow sequence, while `DST_PS_n` and `SRC_PS_n` specifically indicate server-sent and client-sent packets, respectively

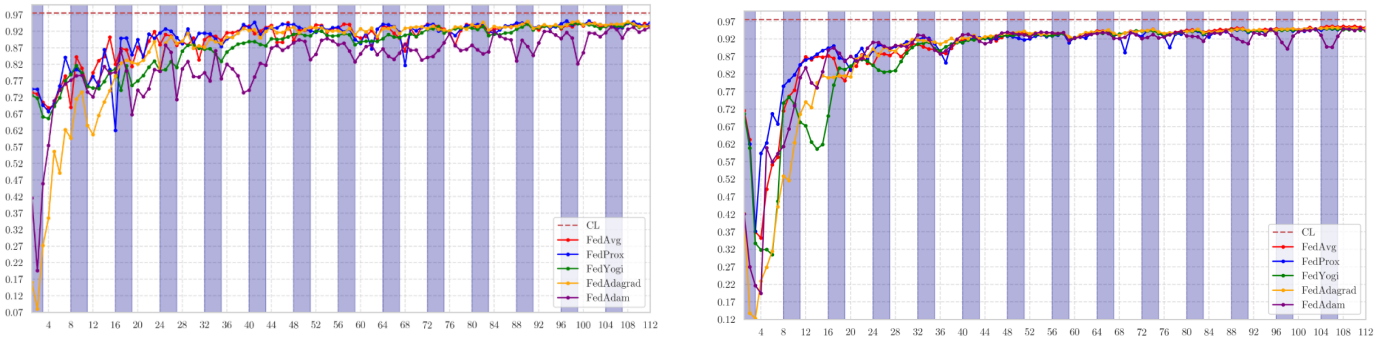
C. Stable and High-Performance Buffered FL

Finally, we evaluate our proposed FL framework with the client-side buffering mechanism enabled. Fig. 3b demonstrates the dramatic improvement. The introduction of the buffer almost completely eliminates the volatility, and all algorithms achieve stable convergence.

FedAvg achieves both the highest average performance and the best stability (lowest standard deviation) through simple weighted averaging based on client data sizes, outperforming more complex adaptive optimizers that proved sensitive to gradient noise. Notably, FedAvg with buffering, achieves a stable F1 score of 95.2%. This represents a performance trade-off of only 2.3 percentage points compared to the non-private centralized model. This suggests that in the presence of non-stationary network traffic, the robustness of a simple average is more beneficial than the complex, and often unstable, updates of adaptive methods like FedAdam. This instability of adaptive optimizers in heterogeneous federated settings is a documented phenomenon, as their momentum-based updates can fail to converge smoothly [16].

V. Conclusion

In this paper, we addressed the critical challenge of classifying encrypted QUIC traffic in a scalable and privacy-preserving manner. We demonstrated that FL, when properly adapted to the unique dynamics of network environments, is a highly effective solution. Our proposed approach, combining the simplicity of the FedAvg algorithm with a novel client-side buffering mechanism, achieves an F1 score of 95.2%.



(a) Without buffering (Standard FL)

(b) With our proposed buffering mechanism

Fig. 3: Comparison of FL convergence behavior over 112 rounds (two weeks).

This represents a remarkably small performance degradation of only 2.3 percentage points compared to an ideal, non-private centralized model.

The experimental results carry practical implications for network operators. The 2.3% performance cost for privacy is a highly favorable trade-off, making collaborative, cross-organizational traffic analysis feasible without compromising data sovereignty. Furthermore, our findings strongly advocate for simplicity in system design; the combination of a basic FedAvg aggregator with a client-side stability mechanism like our buffer provides a robust, effective, and computationally efficient solution for real-world deployment on network monitoring platforms.

Our work opens several avenues for future research. We suggest exploring *1)* asynchronous FL to offer greater client flexibility; *2)* sequential models (e.g., LSTMs) to improve accuracy on long-lived flows; and *3)* adaptive buffering strategies to dynamically optimize the trade-off between data freshness and model stability.

Reproducibility

To ensure the reproducibility of our results and to facilitate future research, all source code, experimental configurations, and data processing scripts are publicly available at our GitHub repository [15]. Beyond the results presented in this paper, the artifact provides extended analyses, metrics, and complementary visualizations, offering deeper insights into per-client performance and the temporal dynamics of the system.

Acknowledgment

Supported by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences. This work was also supported by projects TKP2021-NVA-02 (financed under the TKP2021-NVA scheme) and 2024-1.2.6-EUREKA-2024-00009 (financed under the 2024-1.2.6-EUREKA scheme), implemented with support provided by the Ministry of Culture and Innovation of Hungary from the National Research, Development and Innovation Fund.

References

[1] A. Langley *et al.*, "The quic transport protocol: Design and internet-scale deployment," in *Proceedings of the Conference of the ACM Special Interest Group on Data Communication*, ser. SIGCOMM '17, 2017, pp. 183–196. DOI: 10.1145/3098822.3098842.

[2] V. Tong *et al.*, "A novel quic traffic classifier based on convolutional neural networks," in *2018 IEEE Global Communications Conference (GLOBECOM)*, 2018, pp. 1–6. DOI: 10.1109/glocom.2018.8647128.

[3] J. Luxemburk *et al.*, "Encrypted traffic classification: The quic case," in *2023 7th Network Traffic Measurement and Analysis Conference (TMA)*, 2023, pp. 1–10. DOI: 10.23919/tma58422.2023.10199052.

[4] H. B. McMahan *et al.*, "Communication-efficient learning of deep networks from decentralized data," 2016. DOI: 10.48550/ARXIV.1602.05629.

[5] Z. Jin *et al.*, "Fedetc: Encrypted traffic classification based on federated learning," *Heliyon*, vol. 10, no. 16, e35962, 2024. DOI: 10.1016/j.heliyon.2024.e35962.

[6] J. Luxemburk *et al.*, "Cesnet-quic22: A large one-month quic network traffic dataset from backbone lines," *Data in Brief*, vol. 46, p. 108 888, 2023. DOI: 10.1016/j.dib.2023.108888.

[7] Y. Guo *et al.*, "Wcl: Client selection in federated learning with a combination of model weight divergence and client training loss for internet traffic classification," *Wireless Communications and Mobile Computing*, vol. 2021, no. 1, 2021. DOI: 10.1155/2021/3381998.

[8] Y. Guo and D. Wang, "Feat: A federated approach for privacy-preserving network traffic classification in heterogeneous environments," *IEEE Internet of Things Journal*, vol. 10, no. 2, pp. 1274–1285, 2023. DOI: 10.1109/jiot.2022.3204975.

[9] A. Pekar *et al.*, "Incremental federated learning for traffic flow classification in heterogeneous data scenarios," *Neural Computing and Applications*, vol. 36, no. 32, pp. 20 401–20 424, 2024. DOI: 10.1007/s00521-024-10281-4.

[10] J. Nguyen *et al.*, *Federated learning with buffered asynchronous aggregation*, 2021. DOI: 10.48550/ARXIV.2106.06639.

[11] D. J. Beutel *et al.*, *Flower: A friendly federated learning research framework*, 2020. DOI: 10.48550/ARXIV.2007.14390.

[12] T. Li *et al.*, *Federated optimization in heterogeneous networks*, 2018. DOI: 10.48550/ARXIV.1812.06127.

[13] S. Reddi *et al.*, *Adaptive federated optimization*, 2020. DOI: 10.48550/ARXIV.2003.00295.

[14] S. M. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," in *Proceedings of the 31st International Conference on Neural Information Processing Systems*, ser. NIPS'17, 2017, pp. 4768–4777.

[15] FlowFrontiers, *Federated QUIC Traffic Classification - Digital Artifacts*, https://github.com/FlowFrontiers/FL-QUIC-TC, 2025.

[16] G. A. Baumgart *et al.*, *Not all federated learning algorithms are created equal: A performance evaluation study*, 2024. DOI: 10.48550/ARXIV.2403.17287.