Article

Not peer-reviewed version

FedOps Mobile: A Platform of Federated Learning Management for Enhanced Mobile Collaboration

Farkhod Yusubov and KangYoon Lee [*]

Posted Date: 23 May 2024

doi: 10.20944/preprints202405.1479.v1

Keywords: federated learning; on-device training; system heterogeneity

Article

FedOps Mobile: A Platform of Federated Learning Management for Enhanced Mobile Collaboration

Farkhod Yusubov and KangYoon Lee *

Department of IT Convergence Engineering, Gachon University Seoul, South Korea; farxody98@gachon.ac.kr
* Correspondence: keylee@gachon.ac.kr

Abstract: Federated learning (FL) has emerged as a crucial technology in today's data-centric technological environment, enabling decentralized machine learning and safeguarding user privacy. This study introduces "Federated Operations (FedOps) Mobile," a novel FL framework optimized for the dynamic and heterogeneous ecosystem of mobile devices. FedOps Mobile enhances traditional FL approaches for mobile devices by integrating real-time operational control and advanced on-device training capabilities using TensorFlow Lite and CoreML, addressing critical challenges in scalability, efficiency, and system heterogeneity. Our approach utilizes a wide range of devices, facilitated by intelligent client-selection mechanisms. This mechanism evaluates the capabilities and readiness of multiple devices per client to ensure fair and efficient network participation. The framework also utilizes remote device control for seamless task management and sustained engagement, enabling continuous learning without compromising the user experience. We conducted extensive experiments to validate the framework's performance, focusing on three core aspects: operational efficiency, model personalization, and resource optimization in multi-device environments. The results demonstrate that the proposed method is effective for efficient client selection, energy consumption, and model optimization.

Keywords: federated learning; on-device training; system heterogeneity

1. Introduction

Federated learning (FL) is becoming increasingly crucial in today's technology-driven world, where privacy and efficiency are paramount. This significance is magnified because individuals frequently use multiple personal devices, each capable of contributing to the vast data streams necessary for machine learning. Strategic management of these devices is crucial for effectively leveraging FL, encompassing key aspects such as client selection, model personalization, and comprehensive system control. Furthermore, as the network of participating devices grows and the scale of the system expands, the imperative to manage energy consumption becomes even more critical. This necessity drives the development of sophisticated techniques that ensure that FL systems not only perform well but also optimize energy use, thereby supporting sustainable operations in widespread technological applications. These advancements highlight FL's evolving role as a cornerstone technology that harmonizes data privacy, system efficiency, and sustainability in an increasingly connected world.

The essence of FL lies in its capability to facilitate learning directly at the data source, thereby significantly reducing the risks associated with data transfer and storage at centralized locations. This method is particularly beneficial in scenarios involving sensitive information, such as personal health data or location details, where privacy concerns are paramount. Moreover, FL facilitates the inclusion of a broader diversity of data, reflecting real-world variations more accurately than the data collected under controlled experimental conditions.

The incorporation of a multi-device strategy is particularly advantageous in environments where users frequently interact with several connected devices, such as smartphones, tablets, and wearables. This synergy allows FL to harness richer contextual information, leading to more personalized and accurate models. Moreover, it mitigates the impact of device dropout, a common issue in mobile FL where devices may go offline or become unavailable during the critical stages of the learning process.

Devices in a federated network display significant variability in computing power, storage capacity, battery life, and network connectivity. This heterogeneity can lead to training inefficiencies, resulting in slower convergence rates for less capable devices and balanced contributions across the network. Addressing these disparities is crucial for optimizing the training process and ensuring equitable model development. Effective communication is a major challenge in multi-device environments, especially when considering the potentially large number of devices involved and their varying Internet connectivity. Optimizing the amount of data exchanged and ensuring reliable communication without overloading the network resources are critical concerns.

Selecting the devices to participate in a training session involves balancing multiple factors, such as device availability, data relevance, and current network conditions. Coordinating these devices to effectively participate in the learning process while managing dropout rates (devices unexpectedly going offline) adds another layer of complexity. As the number of devices increases, the complexity of their management also increases. Scalability issues are not just technical but also administrative, as the system must handle thousands of devices without compromising training efficiency and model accuracy. In multi-device environments, managing the energy consumption of the participating devices is crucial. Intensive computational tasks can drain battery life, which may deter users from participating in FL tasks. Developing energy-efficient training algorithms is essential to ensuring sustainable participation. Ensuring the security of the FL process in multi-device environments is challenging but necessary. The system must be robust against malicious attacks, such as data poisoning or model inversion attacks, especially when dealing with multiple devices. As mobile devices continue to evolve and play a more integral role in daily activities, the potential for FL to transform industries—from healthcare to finance and from smart cities to personalized education—is immense. FL not only offers a pathway to more scalable and privacy-conscious machine learning models but also creates opportunities for users to benefit from shared improvements while maintaining control over their data.

In this paper we introduce novel FL framework FedOps Mobile, FedOps was introduces at [1] and [2].

Moon et al. [1] proposed FedOps, a federated learning operations platform designed to manage the entire lifecycle of FL projects, particularly focusing on client heterogeneity and model update aggregation. The FedOps platform enhances traditional machine learning operations (MLOps) to be applicable in FL contexts, addressing the complexity of implementing FL projects by providing tools for deployment, data handling, model handling, and monitoring using cloud-native applications and Kubernetes.

This work was core and server part of FedOps mobile, the FedOps platform provides a comprehensive solution for FL lifecycle management.

Yang et al. [2] proposed FLScalize, a federated learning lifecycle management platform designed to manage the entire lifecycle of FL projects, particularly focusing on system and data heterogeneity simulations. FLScalize extends traditional machine learning operations (MLOps) to be applicable in FL contexts, providing tools for deployment, data handling, model handling, and monitoring using cloud-native applications and Kubernetes. The platform includes manager components for continuous integration, deployment, and training, allowing researchers to easily apply custom data and models to FL environments.

2. Related Works

The evolution of FL technologies has addressed various systemic challenges, from enhancing computational resource management to scaling operations in multi-device environments. In "FLAME: Federated Learning across Multi-device Environments," the study explores a multi-device approach, offering insights into managing device heterogeneity and optimizing computational resources across grouped devices. This methodology aligns with the principles of "federated operations (FedOps)," which further develops these concepts by integrating real-time operations management using TensorFlow Lite and CoreML for on-device training [14].

A further examination of system scalability is presented in "Towards Federated Learning at Scale: System Design," which explores the architectural and operational challenges of scaling FL systems. This foundational work underpins the operational efficiencies and engagement mechanisms implemented in "FedOps" using Firebase for task management and notifications, promoting sustained device engagement in large-scale networks as discussed by Bonawitz et al. [15].

The sector-specific application of FL in "FedHealth: A Federated Transfer Learning Framework for Wearable Healthcare" demonstrates the adaptability of FL to healthcare, emphasizing the optimization of privacy and device capabilities. Similarly, "FedOps" uses device-reported metrics for personalized model adjustments, thus ensuring data privacy and optimized model performance across various applications [3].

Additional discussions on data heterogeneity and optimization are highlighted in "A Survey on Heterogeneous Federated Learning," which aligns with "FedOps" through its implementation of smart selection mechanisms that adapt to the state and capabilities of devices, ensuring efficient learning processes [4].

The concept of FL has evolved as a solution to the challenges posed by the need for privacy-preserving machine learning using decentralized data sources. As demonstrated in the foundational work of McMahan et al. [5], this approach enables collaborative learning among multiple decentralized devices without compromising data privacy.

Efficiency and Scalability Challenges: As FL continues to be adopted on more diverse and larger scales, challenges related to efficiency and scalability have become more prominent. Bonawitz et al. [2] investigated these challenges, emphasizing the necessity for a robust system architecture and efficient communication management to deploy FL at scale. Similarly, Koneˇcny` et al. [6] proposed strategies for improving communication efficiency, which is critical as the number of devices in an FL network grows, to minimize the overhead caused by data transmission during model training.

Security and Privacy Enhancements: In addition to communication efficiency, ensuring the security and privacy of distributed learning processes is crucial. Moriai [7] explored privacy-preserving techniques using additively homomorphic encryption to protect data during the FL process. On the blockchain front, Shayan et al. [8] proposed a ledger that enhances the privacy and security of peer-to-peer machine learning, providing a novel approach to maintaining data integrity and confidentiality in distributed networks.

Healthcare and Mobile Applications: The application of FL in specific domains, such as healthcare and mobile environments, illustrates both the versatility and specialized requirements of FL systems. Chen et al. [3] demonstrated the adaptability of FL in healthcare, highlighting the need for personalized models that can operate on wearable devices. Similarly, Hard et al. [9] demonstrated how FL can be applied to improve mobile keyboard predictions, showcasing the practical benefits of FL in everyday applications.

Advanced Models and Personalization: Addressing data heterogeneity and system heterogeneity, Cho et al. [14] proposed frameworks for effectively managing diverse data and device capabilities within an FL system. These studies underscore the need to develop sophisticated FL models that can handle varying data distributions and device heterogeneity.

Open Problems and Future Directions: Despite considerable advancements, the field of FL continues to burgeon with open problems and research opportunities. Kairouz et al. [4] provided a comprehensive review of the advancements and persisting challenges in FL, offering insights into potential research directions that could further refine the efficiency, security, and applicability of FL systems.

Additional discussions on the complexities and solutions in FL environments were presented by Abhishek et al. [10]. This paper introduces the foundational principles of FL and emphasizes its potential to democratize machine learning through a decentralized approach that enhances data privacy and device autonomy.

The challenges related to data management and optimization strategies in FL were further explored by Blanchard et al. [11]. This study highlights the vulnerabilities of FL systems to adversarial attacks and proposes robust methodologies to enhance the resilience of these systems against malicious participants, ensuring that the integrity of the model training process is maintained.

In real-world applications, Dayan et al. [12] demonstrated the practical implementation of FL in sensitive healthcare environments. This study underscores the importance of FL in scenarios where patient data cannot be centralized due to privacy concerns, showcasing FL's capability to facilitate critical research collaborations without compromising data security.

Furthermore, Reddi et al. [13] detailed advancements in algorithmic efficiency, focusing on the dynamic adjustment of learning rates in federated settings to optimize convergence times and improve the overall efficiency of the learning process. This adaptive approach helps address the diverse conditions and capacities of participating devices, which are crucial for maintaining the performance and speed of FL deployments across heterogeneous networks.

These additional sources and discussions provide a deeper understanding of the evolving landscape of FL, highlighting both the ongoing challenges and innovative solutions that shape the future of decentralized machine learning platforms.

Table 1 provides a comparative overview of some prominent federated learning frameworks, including FedOpsPS Mobile, FedML, Flower, and PySyft, focusing on key features such as cross-platform support, remote device management, energy optimization, supported algorithms, client selection, and background task management.

Table 1. Comparison of Federated Learning Frameworks for mobile.

Feature	FedOpsPS Mobile	FedML	Flower	PySyft
Cross-platform	✓	✓	✓	✓
Remote Device Management	✓	x	x	x
Energy optimization	✓	○	x	○
Supported Algorithms	FedAvg, FedProx, FedYogi	FedAvg, FedProx, personalized FL	FedAvg, FedProx, FedYogi	FedAvg, Secure Aggregation
Client Selection	✓	✓	x	x
Background task managment	✓	✓	x	x

3. FedOps Mobile Overview

In this section, we describe the architecture and components of the proposed framework. The main motivation behind addressing the FedOps mobile-solving system heterogeneity problem is to facilitate the development and implementation of different scenarios. First, we will describe the technologies and frameworks used in this study.

- TensorFlow Lite

TensorFlow Lite is an open-source deep learning framework designed for on-device inference by Google. This allows machine-learning models to run efficiently on mobile and embedded devices. In FedOps, TensorFlow Lite optimizes the models to perform well on low-power and resource-constrained devices, enabling real-time machine learning processing directly on mobile devices. This is critical for maintaining data privacy and leveraging local computational resources.

- CoreML

CoreML is Apple’s machine learning framework that seamlessly integrates with iOS devices to provide optimized performance for model training and execution. In FedOps, CoreML provides efficient and powerful on-device training capabilities for iOS devices, thereby enhancing the overall efficiency and responsiveness of the FL process.

- Firebase

Firebase, developed by Google, offers a variety of tools essential for app development, such as real-time databases, remote configurations, and cloud messaging. For FedOps, Firebase supports background task management and real-time data synchronization, which are crucial for managing the distributed networks of devices in FL scenarios. In our framework, we used Firebase Cloud Messaging (FCM), Firebase Real-Time Database (FRB), and Firebase Cloud Functions (FCF).

- Flutter

Flutter is Google's user interface toolkit for creating visually appealing, natively compiled applications for mobile, web, and desktop devices from a single codebase. This enables FedOps to deliver a consistent and dynamic user interface across various platforms, simplifying user interactions with the FL system.

- Flower mobile Software Development Kit(SDK)

The Flower mobile SDK is a component of the Flower framework designed to facilitate the implementation of FL in mobile environments. FedOps simplifies the deployment and management of FL tasks on mobile devices, ensuring that the system can scale effectively and manage a wide array of device types and capabilities.

FedOps introduces several advancements in FLA, focusing on enhancing training efficiency, operational control, and privacy management across decentralized networks. Key features include:

1. **Cross-Platform On-Device Training:** By integrating TensorFlow Lite and CoreML, FedOps facilitates real-time model training directly on devices, thereby harnessing the native computational capabilities without compromising privacy.

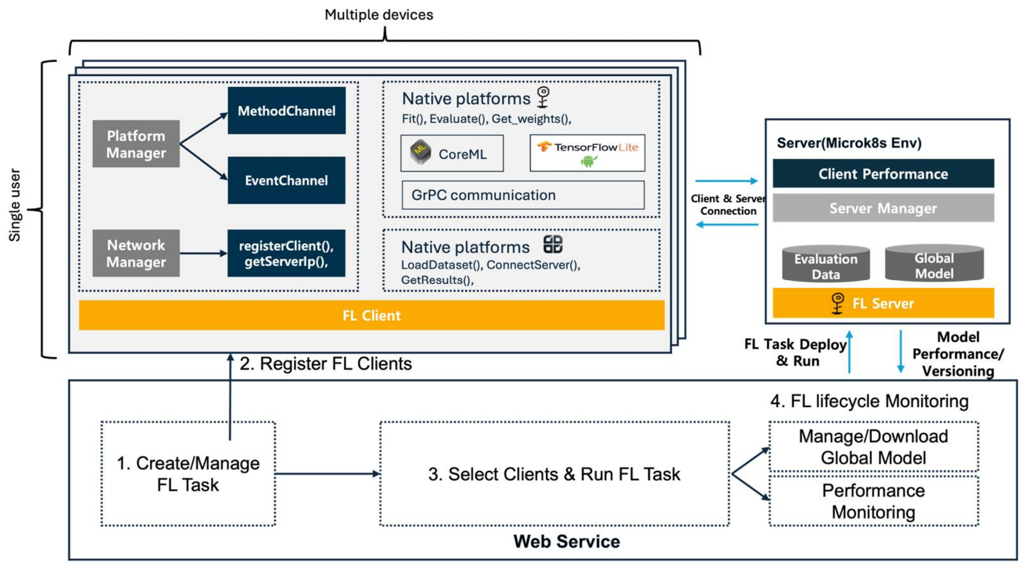

Figure 1. Architecture of FedOps.

2. **Real-Time Operational and User-Device Identification Control:** FedOps utilizes a web platform to provide comprehensive management and monitoring capabilities, allowing for dynamic adjustments and optimizations across the federated network.
3. **Remote Client Selection and Training:** The incorporation of Firebase enables effective background task management and user notifications, which are critical for maintaining engagement and operational efficiency in FL environments.
4. **Adaptive Client Selection:** FedOps employs intelligent client-selection algorithms that evaluate the current state and capabilities of devices, optimize network resources, and ensure uniform model training contributions. Custom client selection methods or changes can be deployed on web platforms.
5. **Personalization and Privacy:** The framework's capability to personalize learning models using aggregated data from a user's device, coupled with stringent privacy protocols, ensures that FedOps is not only effective but also trustworthy.

At the core of the FedOps mobile application, there are two on-device training frameworks: CoreML (iOS) and TFLite (Android). These two frameworks can be integrated through the MethodChannel and EventChannel features of Flutter, which can call native platform functions and monitor continuous tasks such as training. To establish a stable and energy-efficient connection.

The client side of the FedOps system comprises several key components essential for handling on-device machine learning processes and communications.

3.1. Platform Manager

- Manages device-specific functionalities and ensures seamless integration and operation of various system components.
- Facilitates communication with native platform capabilities through MethodChannel and EventChannel.

3.2. Network Manager

- Handles all network-related functionalities, such as registering the client with the FL server and managing network communication.
- Responsible for essential tasks such as registerClient() and getServerIp(), ensuring efficient connectivity with the server.

3.3. Native Platforms

- Implements machine-learning functions such as Fit(), Evaluate(), and Get weights() that are crucial for training and evaluating models on the client device.
- Utilizes TensorFlow Lite and CoreML to optimize machine learning computations on mobile devices.
- Employs gRPC communication for robust and efficient data exchange with the server.

3.4. Server Side (Microk8s Environment)

The server side, hosted in a Microk8s environment, plays a pivotal role in managing and coordinating the FL process.

3.5. FL Server

- Central to the FL process is the aggregation of models and data analysis.
- Stores evaluation data and maintains the global model, updating it with insights derived from client contributions.

3.6. Server Manager

- Manages the overall server operations, including client performance monitoring and data management.
- Coordinates the exchange of data and model updates between the server and clients, thereby ensuring system integrity and performance.

4. Web Service

- Web service represents a component designed to facilitate additional interactions with web-based services or external systems.

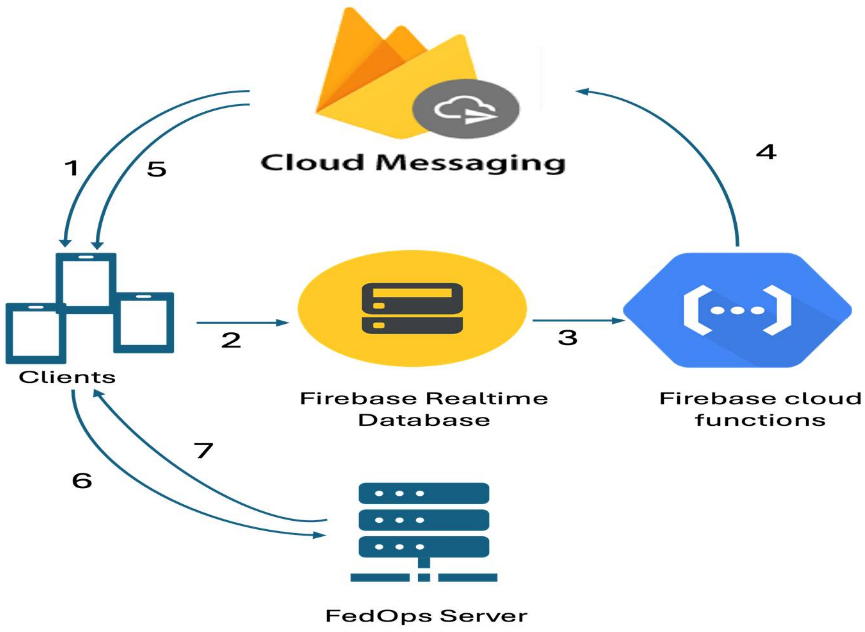

Figure 2. Explanation of how FedOps works after installing the application on mobile. 1. The system administrator registers a new task by sending an FCM message. 2. The online devices send resource information to the FRD. 3. The client selection function is triggered to select devices based on the client selection algorithm 4. Send selected device identifications to the FCM. 5. The FCM sends a new message to call selected devices for training. 6. Selected devices connect to the server to the get initial global model. 7. The server gives the initial global model to the devices.

The FedOps framework enhances the functionality and efficiency of FL systems, with a specific focus on mobile devices. It utilizes a series of cloud-based and local technologies to optimize performance and ensure data privacy. Below, we outline the key operational components and their roles in the system.

4.1. Device Resource Reporting

Upon receiving a notification, the client devices respond by sending critical resource information to a centralized repository. This information includes metrics such as the training and inference accuracy of the last round, current battery level, and communication capabilities. These metrics are crucial for assessing the suitability of each device for participation in the upcoming training rounds.

4.2. Firebase Realtime Database

All device metrics were stored in the Firebase Realtime Database, which serves as a dynamic and responsive platform for managing the large-scale, real-time data needs of FedOps.

4.3. Firebase Cloud Functions and OnWrite Trigger

Utilizing the OnWrite trigger functionality of Firebase Cloud Functions, FedOps automates the client selection process. When new device data is written to the database, these instructions are triggered to evaluate and select the most suitable device for the next training cycle. The selection criteria included device availability, resource status, and past performance metrics, ensuring that only the most capable devices were selected.

4.4. Training Notification

Once the clients are selected, FCM sends a notification to these devices to initiate the training process. This approach ensures that device participation in model training is both timely and resource-efficient, aligned with the availability and capability of each device.

5. Experiments

This section outlines the comprehensive experimental setup designed to evaluate the efficiency, scalability, personalization capabilities, and resource optimization of the FedOps framework in FL environments. We assessed FedOps across various scenarios, emphasizing cross-device FL, model personalization in the presence of data heterogeneity, and handling system heterogeneity.

5.1. Experiment 1: Cross-Device FL Efficiency and Scalability

Objective: Our primary aim was to scrutinize the operational efficiency and scalability of FedOps deployed across multiple devices, each harboring distinct subsets of data, focusing on the effectiveness of the client selection method.

Datasets:

- *MNIST*: This dataset was used to assess basic FL functionalities and initial model personalization.
- *CIFAR-10*: Provided complexity and diversity, testing the framework's scalability and data handling capabilities.
- *FEMNIST*: Ideal for detailed cross-device FL experiments owing to the partitioning by a digit writer.

Methodology:

1. Each dataset was distributed across a simulated network of devices to ensure a non-IID (independent and identically distributed) distribution that mimics real-world scenarios.
2. FedOps was deployed to orchestrate model training by incorporating a client selection algorithm that dynamically chooses devices based on their current state (e.g., battery level, available memory, and network bandwidth).
3. Training sessions were conducted to iteratively refine client selection based on ongoing device performance metrics and resource availability.
4. The performance was compared against baseline FL models and centralized approaches using metrics such as accuracy, loss, and training time efficiency.

5.2. Experiment 2: Model Personalization and Heterogeneity Handling

Objective: To examine the extent to which FedOps supports model personalization and effectively handles heterogeneous data distribution using the client selection method.

Datasets:

- *FEMNIST*: Assesses personalization capabilities.
- *SHL (Sussex-Huawei Locomotion)*: Tests the framework in wearable contexts using diverse human activity data.

Methodology:

1. Personalized models were created for devices using the FEMNIST and SHL datasets, reflecting diverse real-world data conditions.
2. Client selection was implemented to prioritize devices that provide diverse and representative data samples, thereby enhancing the robustness of the global model.
3. The effectiveness of personalization was measured through local and global model accuracy comparisons, and FedOps was evaluated for adaptability to data source diversity through device-specific performance assessments.

5.3. Experiment 3: System Heterogeneity and Resource Optimization

Objective: To assess the performance of FedOps in environments with varying device capabilities by focusing on computational and communication resource optimization.

Datasets:

- The utilization of the CIFAR-10 and SHL datasets caters to the testing of FedOps under different computational demands and real-world mobility scenarios.

Methodology:

1. We simulated a range of devices with varying computational power and network conditions to reflect the broad spectrum of user equipment.
2. Client selection was integrated to optimize resource allocation and prioritize devices based on computational efficiency and network stability.
3. We applied model compression techniques and adaptive communication protocols tailored to the capabilities of the selected devices.
4. The impact of these optimizations on the training duration, model accuracy, and communication overhead was evaluated and compared with traditional FL setups.

5.4. Data Privacy and Security Evaluation

Objective: Through qualitative analysis, we aimed to dissect and document the inherent data privacy and security features of FedOps, especially in comparison with existing FL frameworks.

Methodology:

1. The setup of each experiment, including device distribution, FL parameters, and specific FedOps configurations, was meticulously documented.
2. Multiple iterations were conducted to ascertain the statistical significance, employing metrics suitable for each objective, such as model accuracy, precision, and recall, along with efficiency indicators, such as training duration and data consumption.
3. Ablation studies were conducted to determine the impact of distinct FedOps components on the overall performance of the framework.

This experimental framework was designed to offer a panoramic understanding of the capabilities and limitations of FedOps, thereby paving the way for informed enhancements and applications in the FL domain.

5.5. Device Configuration for Experiments Overview

Our experiments were conducted using a diverse array of devices to simulate real-world conditions, and ensure that our findings are representative of various user environments. This approach not only strengthens the validity of our results but also tests the robustness of the FedOps framework across different hardware configurations.

Devices Used

- *Lab Devices*: Two real mobile devices available in our laboratory, specifically tablets, were used because of their enhanced screen size and processing capabilities suitable for intensive FL tasks.
- *Emulators*: Two emulators were configured to represent average consumer smartphones, creating a controlled environment for testing and debugging specific scenarios.
- *AWS Device Farm*: Twenty-one devices were accessed remotely via the AWS Device Farm, which provides a broad range of mobile devices with various operating systems, screen sizes, and hardware capabilities. This selection included both high-end smartphones and mid-range models to cover a wide spectrum of device capabilities.

Selection Criteria

The devices were selected randomly from the AWS Device Farm to encompass a wide variety of hardware specifications, including different processors, memory capacities, and storage options. This randomness in selection helps mimic the unpredictability of device availability in typical FL scenarios, where each participant may have different device characteristics.

Impact on Experimentation

The heterogeneity of devices ensures that our experiments account for a realistic range of performance metrics such as computational power and network connectivity, which are critical for assessing the scalability and efficiency of FL systems. This setup mimics a realistic FL environment where different devices with varying capabilities contribute to the learning process.

Table 1. Different experiment evaluation results.

Experiment	Device Numbers	Training Rounds, Epoch	Training Time (Overall, Avg Round)	Energy Consump-tion (% of Max Battery)	Accuracy of Global Model	Best Accuracy of Devices
Efficiency, Scalability, and Client Selection were Objectives. Model Personalization and Heterogeneity	25 25 25	30, 5 30, 5 30, 5	2 h, 5 min 1.5 h, 6 min 2.5 h, 7 min	20% 16% 22%	60% 54% 67%	85% 61% 58%

Handling System. Heterogeneity and Resource Optimization.	

6. Conclusion

This study introduces the FedOps framework, a pioneering approach designed to address the critical challenges inherent in FL, particularly focusing on cross-device scenarios, data and system heterogeneity, and resource optimization. Through a comprehensive set of experiments, we demonstrated the capability of FedOps to efficiently and effectively manage the complexities of decentralized learning environments, thereby making significant strides towards realizing the full potential of FL.

Our experimental results underscore the efficacy of FedOps in enhancing the model performance across a diverse array of datasets, including MNIST, CIFAR-10, and FEMNIST, under varying data distribution and system capability conditions. Notably, FedOps demonstrated remarkable proficiency in handling non-IID data and system heterogeneity, thereby ensuring personalized model performance and resource-efficient FL. These achievements mark a significant advancement in the FL domain, showcasing the potential of FedOps to facilitate scalable, efficient, and privacy-preserving machine learning across myriad devices.

Moreover, the FedOps framework illustrates its robustness in real-world applications, particularly in wearable devices and mobile sensing, where data privacy and computational constraints are paramount. By leveraging adaptive optimization techniques and privacy-enhancing protocols, FedOps sets a new standard for FL frameworks, offering a viable solution to the pressing challenges of data privacy, security, and efficient computation in distributed learning environments.

Author Contributions: Design and writing, F.Y; conceptualization, F.Y. and K.L.; software, F.Y.; writing—original draft preparation, K.L.; supervision and contribution, K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported in part by a Commercialization Promotion Agency for R&D Outcomes (COMPA) grant funded by the Korean government (MSIT) under Grant 2022-Future Research Service Development support-1-SB4-1, and in part by a National Research Foundation of Korea (NRF) grant funded by MSIT under Grant NRF-2022R1F1A1069069.

Data Availability Statement: No new data were created or analyzed in this study. Data sharing is not applicable to this article.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Moon, J., Yang, S., & Lee, K. (2024). FedOps: A Platform of Federated Learning Operations with Heterogeneity Management. IEEE Access. DOI: 10.1109/ACCESS.2024.3349691.
2. [1] S. Yang, J. Moon, J. Kim, K. Lee, and K. Lee, "FLScalize: Federated Learning Lifecycle Management Platform," IEEE Access, vol. 11, pp. 47212-47222, 2023, doi: 10.1109/ACCESS.2023.3275439.
3. Chen, Y.; Qin, X.; Wang, J.; Yu, C.; Gao, W. Fedhealth: A federated transfer learning framework for wearable healthcare. *IEEE Intell. Syst.* **2020**, *35*, 83–93. DOI:10.1109/MIS.2020.2988604.
4. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M., Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Foundations and trends® in machine learning* **2021**, 14(1–2), 1–210.
5. McMahan, B.; Moore, E.; Ramage, D.; Hampson, S.; y Arcas, B.A. Communication-efficient learning of deep networks from decentralized data. *Artificial intelligence and statistics*. In *PMLR*, **2017**, 1273–1282.
6. Koneˇcny`, J.; McMahan, H.B.; Ramage, D.; Richtárik, P. Federated optimization: Distributed machine learning for on-device intelligence. *arXiv preprint arXiv:1610.02527*, **2016**.
7. Moriai, S. Privacy-preserving deep learning via additively homomorphic encryption. In *26th Symposium on Computer Arithmetic (ARITH); IEEE Publications*; IEEE Publications, **2019**; pp. 198–198. DOI:10.1109/ARITH.2019.00047.

8. Shayan, M.; Fung, C.; Yoon, C.J.M.; Beschastnikh, I. Biscotti: A ledger for private and secure peer-to-peer machine learning. *arXiv Preprint ArXiv:1811.09904*, **2018**.
9. Hard, A.; Rao, K.; Mathews, R.; Ramaswamy, S.; Beaufays, F.; Augenstein, S.; Eichner, H.; Kiddon, C.; Ramage, D. Federated learning for mobile keyboard prediction. *arXiv preprint arXiv:1811.03604*, **2018**.
10. Abhishek, A.; Binny, S.; Johan, R.; Nithin, R.; Vishal, T. Federated learning: Collaborative machine learning without centralized training data. *Int. J. Eng. Technol. Manag. Sci.* **2022**, *6*, 355–359.
11. Blanchard, P.; Mahdi El Mhamdi, E.; Guerraoui, R.; Stainer, J. Machine learning with adversaries: Byzantine tolerant gradient descent. *Adv. Neural Form. Process. Syst.* **2017**, *30*.
12. Dayan, I.; Roth, H.R.; Zhong, A.; Harouni, A.; Gentili, A.; Abidin, A.Z.; Liu, A.; Costa, A.B.; Wood, B.J.; Tsai, C.-S.; et al. Federated learning for predicting clinical outcomes in patients with Covid-19. *Nat. Med.* **2021**, *27*, 1735–1743. DOI:10.1038/s41591-021-01506-3.
13. Reddi, S.; Charles, Z.; Zaheer, M.; Garrett, Z.; Rush, K.; Kone˘cny`, J.; Kumar, S.; McMahan, H.B. Adaptive federated optimization. *arXiv Preprint ArXiv:2003.00295*, 2020.
14. Cho, H.; Mathur, A.; Kawsar, F. Flame: Federated learning across multi-device environments. *Proc. ACM Interact. Mob. Wearable Ubiquitous Technol.* **2022**, *6*, 1–29. DOI:10.1145/3550289.
15. Bonawitz, K.; Eichner, H.; Grieskamp, W.; Huba, D.; Ingerman, A.; Ivanov, V.; Kiddon, C. Jakub Kone˘cny`, Stefano Mazzocchi, Brendan McMahan, et al. Towards federated learning at scale: System design. *Proceedings of the Machine Learning and Systems* **2019**, *1*, 374–388.