Online-Journals.org

iJOE International Journal of Online and Biomedical Engineering

iJOE | eISSN: 2626-8493 | Vol. 20 No. 14 (2024) | OPEN ACCESS

https://doi.org/10.3991/ijoe.v20i14.50559

PAPER

AdaptiveMesh: Adaptive Federated Learning for Resource-Constrained Wireless Environments

Lamir Shkurti[1](✉), **Mennan Selimi**[1,2]

[1]Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia

[2]Max van der Stoel Institute, South East European University, Tetovo, North Macedonia

ls29773@seeu.edu.mk

ABSTRACT

Federated learning (FL) presents a decentralized approach to model training, particularly beneficial in scenarios prioritizing data privacy, such as healthcare. This paper introduces *AdaptiveMesh*, an FL adaptive algorithm designed to optimize training efficiency in heterogeneous wireless environments. Through dynamic adjustment of training parameters based on client performance metrics, including central processing unit (CPU) utilization and accuracy trends, *AdaptiveMesh* aims to enhance model convergence and resource utilization. Experimental evaluations on heterogeneous client devices demonstrate the algorithm's effectiveness in improving model accuracy, stability, and training efficiency. Results indicate a significant impact on CPU adaptation in preventing client overloading and mitigating overheating risks. Furthermore, the results of the one-way analysis of variance (ANOVA) and regression analysis highlight significant differences in CPU usage, accuracy, and epochs between devices with varying levels of hardware capabilities. These findings underscore the algorithm's potential for practical deployment in real-world edge computing environments, addressing challenges posed by heterogeneous device capabilities and resource constraints.

KEYWORDS

adaptive federated learning (FL), embedded machine learning, wireless mesh networks

1 INTRODUCTION

Federated learning (FL) has emerged as a promising paradigm for decentralized model training, particularly in scenarios where data privacy is important, such as healthcare [1]. FL facilitates collaborative model training across distributed clients while centralizing model updates on a server. This approach addresses data privacy concerns by keeping sensitive data local to the clients [2]. While FL offers significant advantages in terms of data privacy and decentralized training, optimizing its performance in heterogeneous environments remains a challenge [3]. In such environments, the data available to each edge device contributing to the model can vary in quantity and quality, depending on factors such as device location and individual

Shkurti, L., Selimi, M. (2024). AdaptiveMesh: Adaptive Federated Learning for Resource-Constrained Wireless Environments. *International Journal of Online and Biomedical Engineering (iJOE)*, 20(14), pp. 22–37. https://doi.org/10.3991/ijoe.v20i14.50559

Article submitted 2024-06-24. Revision uploaded 2024-07-29. Final acceptance 2024-08-23.

user patterns (e.g., different numbers of training samples). Additionally, the processing power of these resource-constrained devices can differ based on their hardware capabilities or if they are running other applications simultaneously. In a wireless environment, the limitations in network bandwidth, either because of the device's location or due to dynamic network conditions, as well as battery consumption, may cause delays in how fast the overall model learns [4], [5].

The importance of addressing these challenges lies in the increasing deployment of FL in real-world applications, particularly in healthcare, where data privacy and resource optimization are crucial. Effective FL implementations can lead to improved healthcare outcomes by enabling the use of sensitive patient data without compromising privacy. This study proposes the *AdaptiveMesh* algorithm, designed to optimize training efficiency in heterogeneous wireless environments by dynamically adjusting training parameters based on client performance metrics, aiming to improve efficiency and accuracy in such settings [6], [7].

The objective is to develop an adaptive design for the FL clients that leverages the inherent heterogeneity of the participating devices (CPU, memory) and wireless network (bandwidth, etc.). [8], [9]. Specifically, this study employs an adaptive mechanism for FL clients to assess whether adjustments to the training parameters (training images, model accuracy) are necessary. To optimize training efficiency, the adaptive algorithm continuously monitors accuracy, CPU, and memory utilization between rounds. It analyzes trends in both model performance (accuracy history) and CPU usage to determine if adjustments are necessary. If detects issues, like performance stagnation related to the accuracy or rising CPU usage above a predefined threshold, intervenes by initially reducing the training images. This adaptive mechanism, by dynamically aligning resource allocation with the model's evolving needs, ensures that training progresses efficiently among the clients.

The main contributions of the paper are:

1. To exploit the heterogeneous environment of clients, this study investigates the design and development of the *AdaptiveMesh*-adaptive FL algorithm that determines at each round the workload capacity of each client for model training in a determined time slot. The adaptive behavior of *AdaptiveMesh* calculates the difference in CPU utilization between consecutive training rounds and assesses model performance and CPU utilization trends to determine if adaptation is required.
2. The proposed adaptive FL algorithm is evaluated by deploying it on real-world, heterogeneous low-capacity devices integrated within a wireless mesh network.

To better understand the impact and effectiveness of the *AdaptiveMesh* algorithm, this study aims to address the following research questions:

1. How does algorithm optimization enhance FL efficiency in heterogeneous wireless environments?
2. What impact does algorithm optimization have on CPU utilization, network bandwidth, and model accuracy in resource-constrained devices?
3. How does the adaptive adjustment of training parameters based on client performance metrics improve model convergence and resource utilization in FL?

2 RELATED WORK

In this section, existing studies on adaptive FL for devices with limited capacity are reviewed, and methods to protect data privacy are highlighted, which is crucial when handling confidential personal health information during training.

The study FedTCR [10] addresses communication inefficiencies and straggler problems in FL by constructing logical computing clusters and implementing an intra-cluster collaborative training mechanism. This reduces communication costs and ensures even the slowest devices contribute effectively to the training process. While FedTCR focuses on reducing communication overhead and balancing participation among devices, *AdaptiveMesh* emphasizes real-time adaptability and resource management. This ensures optimal resource usage and prevents overheating, thus improving model convergence and stability across heterogeneous devices.

Similarly, FedAdapt: Adaptive offloading for IoT devices in FL by [11] introduces an adaptive offloading framework designed to enhance local training efficiency by offloading deep neural network (DNN) layers to servers. FedAdapt uses reinforcement learning-based optimization and clustering techniques to dynamically identify the optimal layers for offloading for each device. This approach significantly reduces training time and adapts to fluctuating network conditions. While FedAdapt focuses on offloading DNN layers to servers to reduce training time and adapt to network conditions, *AdaptiveMesh* extends this by dynamically adjusting training parameters like the number of images and epochs based on real-time client performance metrics such as CPU utilization and accuracy trends. This additional adaptability helps prevent client overloading and overheating, ensuring optimal resource usage and improving model convergence.

Another study [12] introduces Fed-RAC, which uses resource-aware clustering to optimize training and communication efficiency among diverse devices. By dynamically determining the optimal number of clusters using Dunn Indices, Fed-RAC adapts to varying heterogeneity levels, ensuring efficient model training and aggregation. It employs a master-slave technique through knowledge distillation, enhancing the performance of lightweight models within clusters. While Fed-RAC focuses on resource-aware clustering to optimize training among heterogeneous devices, *AdaptiveMesh* adapts training parameters dynamically based on client performance metrics to improve resource utilization and model convergence. Fed-RAC enhances low-configuration models using master-slave techniques, whereas *AdaptiveMesh* prevents client overload by real-time parameter adjustments.

Furthermore, [13] introduces FL as a communication-efficient approach to training deep neural networks over distributed sources of data while preserving data privacy, which is foundational for collaborative model training in distributed environments.

Regarding adaptive behavior in FL systems, an optimization algorithm tailored for FL systems is proposed in [14]. The work dynamically adjusts learning rates and other hyper-parameters based on client characteristics and network conditions, leading to improved model convergence and performance. This study also takes into consideration network parameters such as bandwidth.

Another significant contribution is [15], which addresses leveraging data from multiple medical institutions while ensuring privacy and security. The study introduces adaptive client sampling (ACS), and the authors aim to optimize the selection of clients for model training in each iteration. This method considers client performance and data characteristics, enhancing the efficiency and effectiveness of the FL process. The research contributes to the ongoing efforts in developing efficient and scalable ML solutions for healthcare, ultimately improving patient care while safeguarding sensitive medical data. In this study, healthcare data consisting of chest X-ray images are utilized.

Moreover, a novel approach [16] proposes an adaptive distribution of resources for cost-effective FL at the wireless network edge, which ensures minimal latency and high performance when learning. By leveraging Lyapunov stochastic optimization techniques, the authors dynamically optimize radio parameters and computation

resources to strike the best balance between energy consumption, latency, and learning performance. The recommended method is customized to federated least mean squares (LMS) estimation, and numerical experiments indicate that it is effective in providing cost-effective, minimal latency federated machine learning at the wireless network edge.

Another study [17] introduces an approach to enhance FL through adaptive client-side hyperparameter optimization. The key elements of their approach include dynamic adjustment: hyper-parameters such as learning rate and batch size are dynamically adjusted for each client based on their computational resources and data characteristics.

In addition, RingSFL [18] proposes an adaptive FL system designed for environments with heterogeneous clients, similar to our environment. The system addresses the challenges posed by varying data distributions and performance capabilities among clients through innovative methodologies. RingSFL leverages a ring-based architecture to organize clients hierarchically, facilitating efficient communication and coordination within a distributed network. The adaptive client sampling technique dynamically selects clients based on their data characteristics and performance metrics, ensuring optimal participation in global model training. In terms of communication optimization and data privacy, clients in this study participate in the training process based on their CPU, memory, and bandwidth resources.

In the context of medical images, [19] addresses the challenges of FL applied to medical image analysis. They propose FedRSMax, a novel aggregation method that enhances the collaborative model aggregation process in FL scenarios. The key innovation of FedRSMax lies in its dynamic aggregation strategy, which optimizes the contributions of locally trained models from distributed medical institutions. Unlike conventional FL approaches that use uniform averaging or simple aggregation methods, FedRSMax dynamically adjusts aggregation weights based on the performance metrics of each local model. This adaptive weighting mechanism ensures that more accurate and reliable models contribute proportionally more to the final aggregated model.

Additionally, [20] introduce a novel approach to FL specifically tailored for wireless mesh networks. The authors address the challenges associated with training ML models in decentralized environments with limited resources and intermittent connectivity. They propose a server-side adaptive mechanism where the central server dynamically adjusts the training process based on network conditions and client capabilities. This adaptive approach optimizes resource utilization and enhances model convergence in wireless mesh network settings. The authors demonstrate the effectiveness of their approach in improving model convergence and reducing communication overhead.

Recent studies mentioned above have explored adaptive algorithms to optimize performance. However, these approaches do not adequately consider the variability in network conditions and device capabilities. In addition to the concepts presented in these studies, the research focuses on adapting to node heterogeneity through dynamic workload capacity adjustments. This involves allocating more work to faster nodes, aiming to improve the overall performance of the global model.

3 *ADAPTIVEMESH* ALGORITHM

The *AdaptiveMesh* algorithm highlighted in Algorithm 1 optimizes FL by dynamically adjusting training parameters based on client performance metrics. *AdaptiveMesh* algorithm is initially configured with key parameters such as global and local learning rates, number of epochs, number of images, batch size, and other

training-related parameters. The following steps outline the key components of the algorithm:

1. **Client registration:** Initially, the clients are registered within the system to participate in the collaborative training process.
2. **Initial configuration:** The server distributes initial configurations to heterogeneous clients (*min_epochs* = 1). Prior to each training round, a time limit is set to ensure training does not exceed a predefined duration.
3. **Training and monitoring metrics:** During each round, CPU utilization, memory utilization, and accuracy metrics are monitored and stored for subsequent analysis.
4. **Adaptive mechanism activation:** After three rounds of training (*round* > 3), an adaptive mechanism is employed to assess whether adjustments to the training parameters are necessary.
5. **Adaptation process:** The adaptive part calculates the difference in CPU utilization between consecutive training rounds and assesses model performance and CPU utilization trends to determine if adaptation is required. The minimum number of training images is six (*min_images* = 6).
   a) When adaptation is required (case: resources are available), and resources are available, a threshold-based approach is employed. For example, if the CPU utilization is below 80%, the number of trained images is adjusted accordingly, with the maximum number of sample images potentially increasing to 234.
   b) When adaptation is required (case: resources are not available), i.e., the trained images are decreased in the clients having CPU utilization > 80%.

The adaptive decision-making process of the *AdaptiveMesh* algorithm is crucial for optimizing model performance across heterogeneous clients in wireless mesh environments. This adaptive mechanism guarantees that resource allocation is dynamically adjusted at the clients in order to meet the evolving demands of model training. Also, the way the algorithm sets up parameters for FL shows a thoughtful way of coordinating shared learning, which is important for improving distributed machine learning in practical applications.

Algorithm 1: AdaptiveMesh: ***Client-side Adaptive Federated Learning for Resource-Constrained Wireless Environments***

```
Require : round, cpu_limit, min_epochs, min_images, max_images, accuracies[],
          cpu_history[], client_images[], client_epochs[]
Result  : new_epochs, new_images
i ← round;
if i > 3 then
    delta_cpu1 ← cpu_history[i − 1] − cpu_history[i − 2] ;
    delta_cpu2 ← cpu_history[i − 2] − cpu_history[i − 3] ;
    diff ← delta_cpu2 − delta_cpu1 ;
    if (accuracies[−1] > accuracies[−2]) AND (accuracies[−1] < 100) then
        if (diff ≤ 0) AND (cpu_history[i − 1] ≥ cpu_limit) then
            new_images ← max(client_images[−1] − 10, min_images) ;
            new_epochs ← max(client_epochs[−1] − 1, min_epochs) ;
        else
            new_images ← min(client_images[−1] + 10, max_images) ;
            new_epochs ← client_epochs[−1] + 1 ;
    else
        if (cpu_history[i − 1] ≤ cpu_limit) then
            new_images ← min(client_images[−1] + 10, max_images) ;
            new_epochs ← client_epochs[−1] + 1 ;
        else
            new_images ← max(client_images[−1] − 10, min_images) ;
            new_epochs ← max(client_epochs[−1] − 1, min_epochs) ;
fl_config.epochs ← new_epochs ;
fl_config.training_images ← new_images;
```

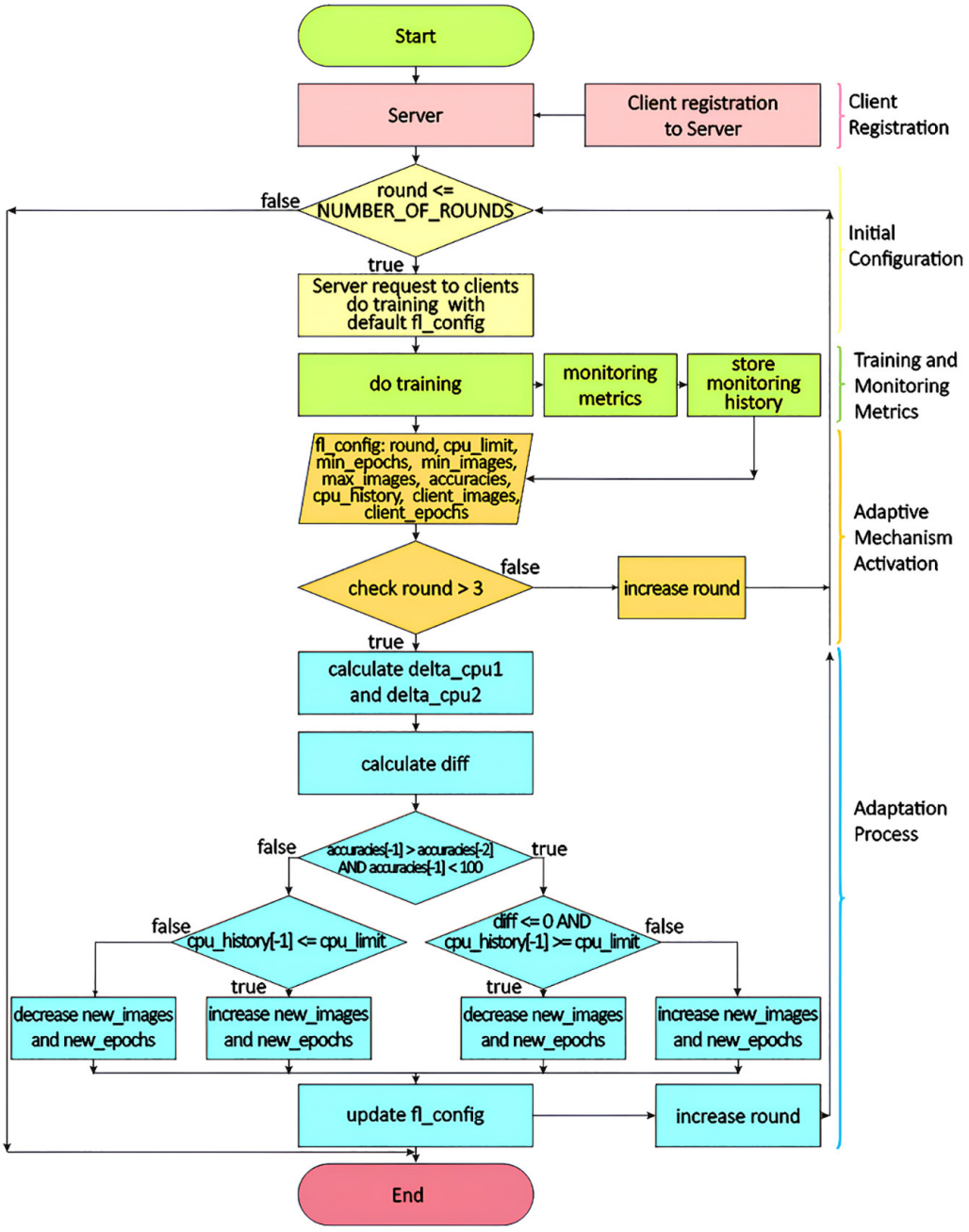

Fig. 1. *AdaptiveMesh* algorithm flowchart

4 EXPERIMENTAL SETUP AND RESULTS

4.1 Experimental setup

The experimental setup includes physical IoT devices interconnected via a wireless network. Figure 2 highlights the wireless testbed used for the experiments. Two types of nodes are identified: the server and clients. The server acts as the central orchestrator, which distributes initial configurations to clients. As a server, use a laptop with an Intel(R) Core (TM) i5-8250U CPU, equipped with 8 GB of RAM, running

the Windows 11 operating system (OS). The clients consist of four distinct devices: 1) a MiniPC NEO Z83-4 device featuring an Intel Atom x5-Z8350 CPU, 4 GB DDR3 RAM, and Windows 10 OS, 2) a laptop with an Intel(R) Core (TM) i5-1035G1 CPU, 8 GB RAM, and Windows 11 OS, and 3) two Raspberry Pi 4 Model B units, namely RPI4 and RPI5, each equipped with a Quad Core Cortex-A72 CPU, 8 GB RAM, 128 GB storage, and running the Linux Ubuntu 22.04.4 LTS OS. In this setup, the server orchestrates the FL process issuing instructions to the client devices. After each training round, the server combines the knowledge gathered from each client to create a collective learning output. The server sends this output back to the clients along with instructions for the next learning round. Clients autonomously regulate their workload acceptance thresholds (see Algorithm 1), thus deviating from the conventional uniform instruction dissemination paradigm.

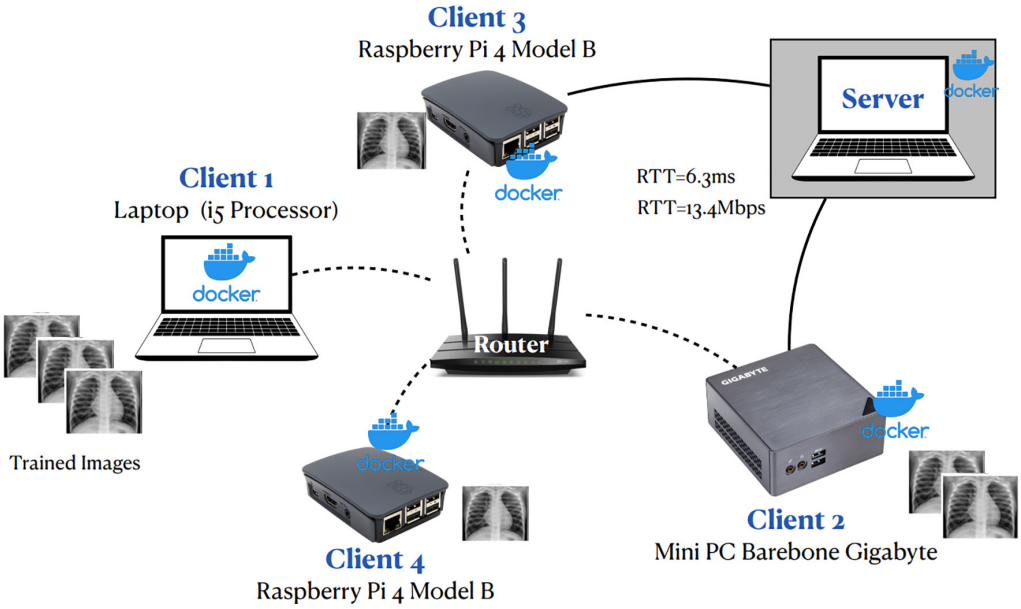

Fig. 2. Testbed nodes used for the experimentation

4.2 Dataset

For the experiments, this study used a dataset of 5,863 JPEG X-ray images from [21]. The images are divided into two categories: pneumonia and normal. These images were taken from young patients aged one to five years old at the Guangzhou Women's and Children's Medical Center in China. The objective is to train a 6-layer convolutional neural network (CNN) using this chest X-ray dataset for the adaptive FL task.

4.3 Experiments

The experimentation assesses how the adaptive client design affects FL in a real-edge wireless environment. Initially, CPU and memory utilization experiments are conducted to demonstrate the system's adaptive behavior. Subsequently, accuracy and loss tests are performed, illustrating the variation in the number of images trained on each device following the application of adaptive behavior. The final section discusses the findings.

4.4 Results

Figure 3 shows the CPU utilization of the four different clients. Figure 3 reveals that model training necessitates substantial computational power. However, the algorithm's CPU adaptation has led to optimization in training. Upon the CPU usage surpassing 80%, the algorithm optimizes the training process for clients by reducing computational loads in the next rounds, thus preventing client overburdening.

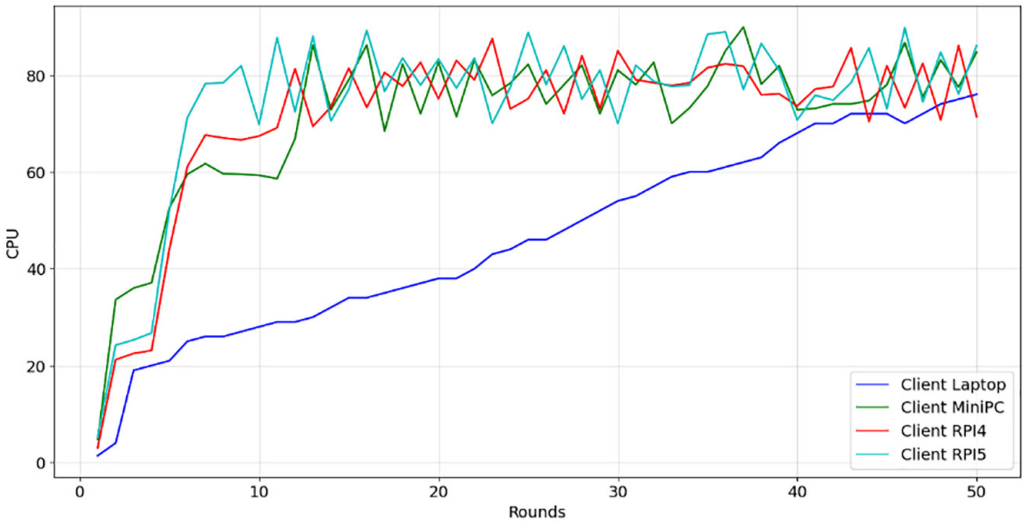

Fig. 3. CPU utilization in different training rounds

Figure 4 shows the RAM memory utilization for different clients. Figure 4 indicates that clients managed memory demands well, utilizing it relatively steadily across different rounds.

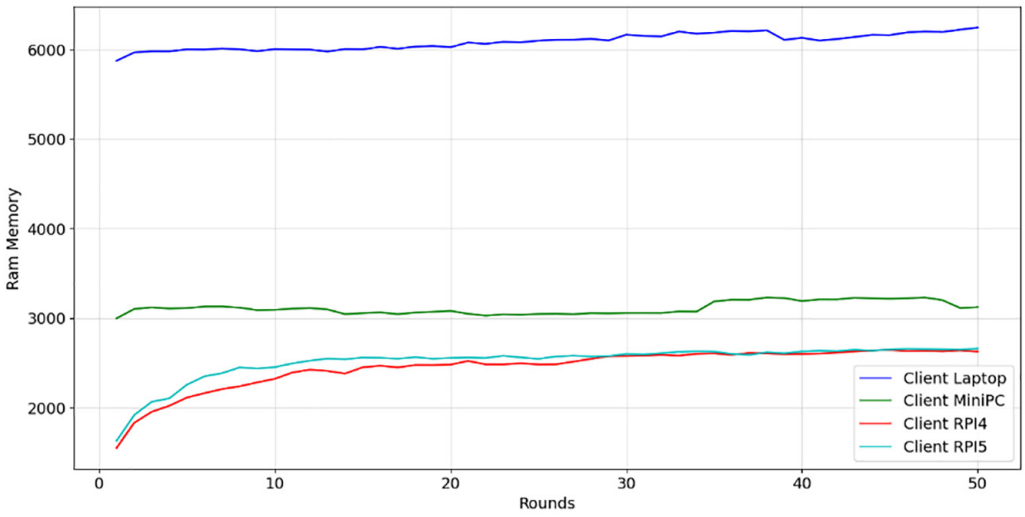

Fig. 4. Memory utilization in different training rounds

Figure 5 illustrates the comparison of achieved accuracies during training across different rounds. It is observed that as the rounds increase, the accuracy also

increases until round 50. Specifically, the laptop achieves 98% accuracy, MiniPC 83%, RPI4 82%, and RPI5 79%.

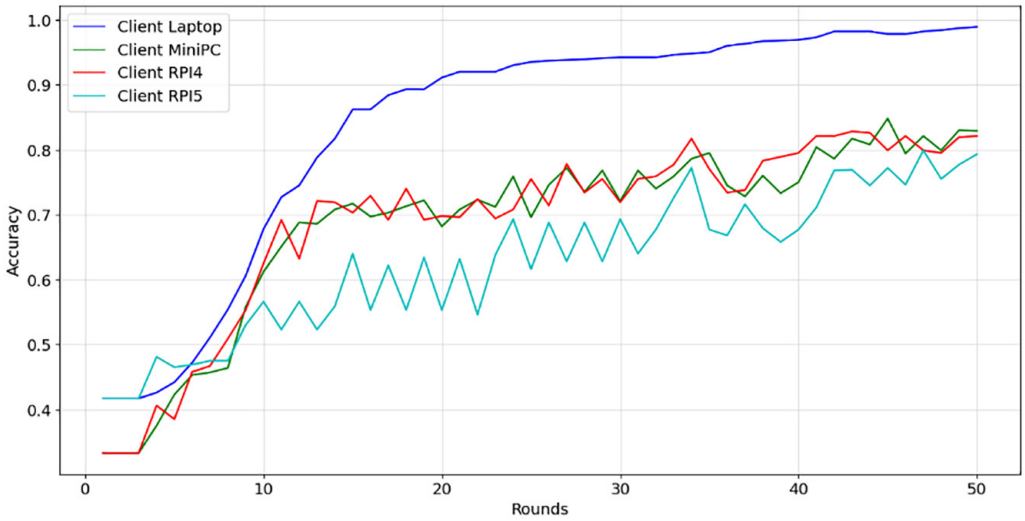

Fig. 5. Accuracy in different training rounds

Figure 6 indicates that clients initially experience high losses, which gradually decrease as the rounds increase. This shows that model training is improving over time.

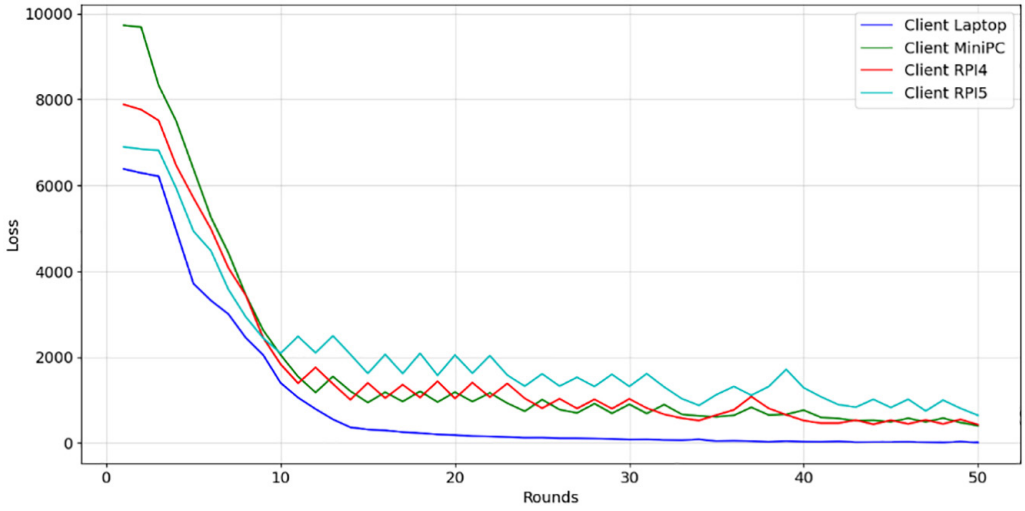

Fig. 6. Losses in different training rounds

Figure 7 shows the number of epochs (local iterations) performed in each round by different clients. Figure 7 reveals that epochs increase gradually as rounds increase. After round 10, in clients reaching the threshold of their CPUs (e.g., RPI4, RPI5), the number of epochs decreases due to the adaptive behavior of the algorithm, i.e., devices with more CPU resources (laptop, miniPC) tend to perform more epochs during training, and vice versa. By round 50, the laptop client reaches 48 epochs, the MiniPC reaches 18 epochs, the RPI4 reaches 16 epochs, and the RPI5 reaches 12 epochs.

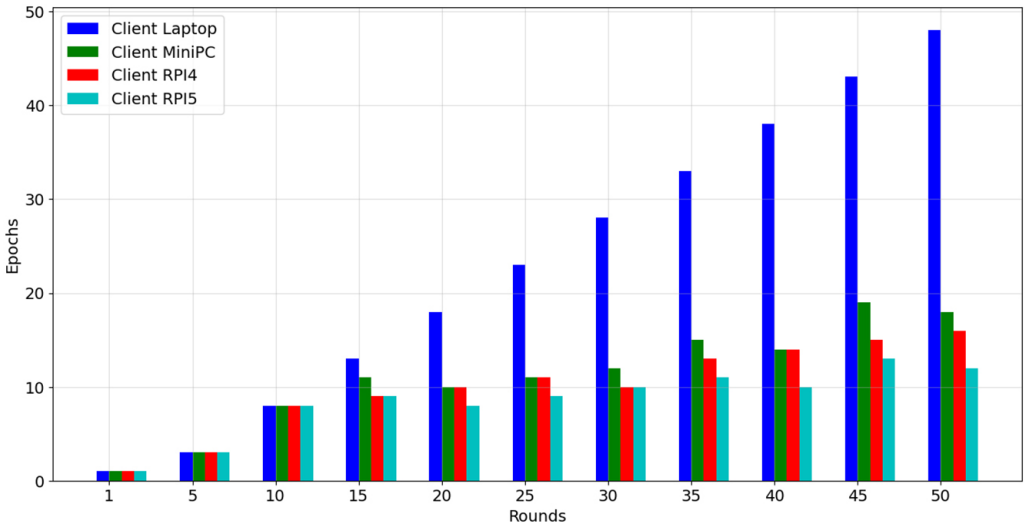

Fig. 7. The number of epochs for various clients reached in different training rounds

Regarding the trained images per client, Figure 8 shows that the number of images increases gradually as rounds increase. However, after round 10, some clients with lower performance cannot cope with the workload of image samples and epoch increase, as their CPU usage exceeds 80%, consequently, these clients request less workload, thereby monitoring CPU performance. When the CPU usage falls below 80% they request an increase in the number of images. Conversely, the laptop client, with higher capacity, consistently demands an increase in the number of images, reaching the maximum allowable values.

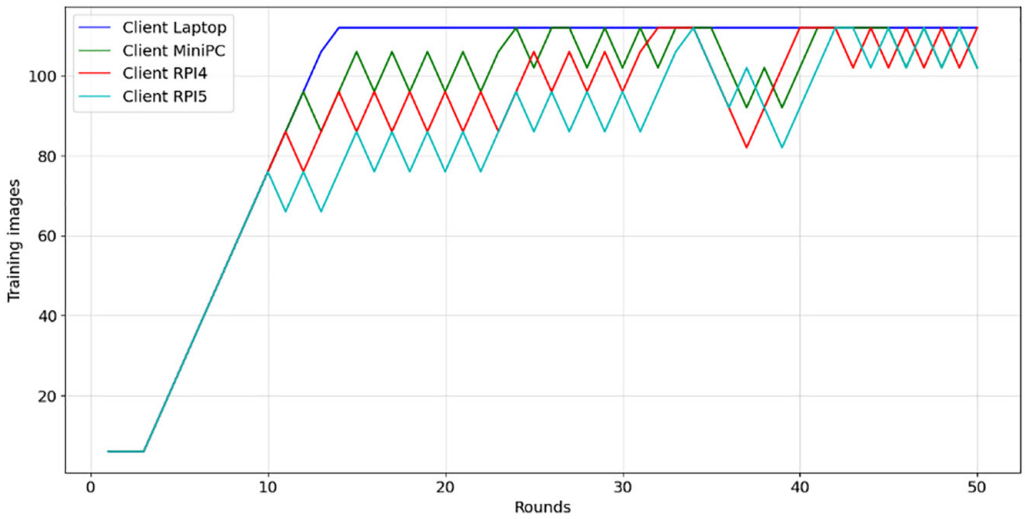

Fig. 8. The number of trained images per client

Figure 9 reveals that the training time for clients increases after the third round due to the beginning of their adaptation, where each client requires an increase in the number of training samples and the number of epochs according to their workload capacity. Also, it can be seen that the training time to time increases and sometimes decreases due to the loading of clients based on their CPU usage. When the CPU utilization is above 80%, clients reduce the number of samples and epochs, leading to reduced training time and reduced computational load that helps prevent device overheating.

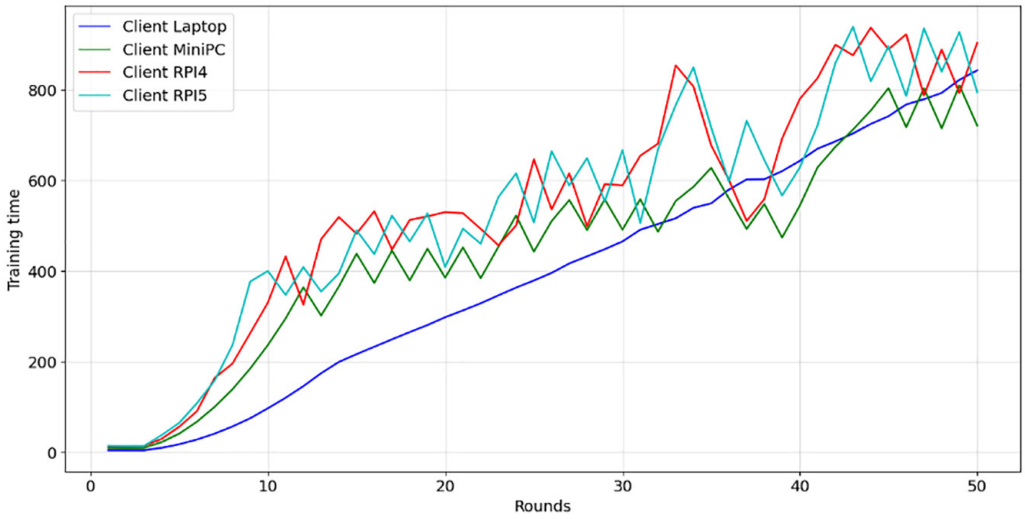

Fig. 9. Training time increases as adaptation begins, adjusting image samples/epochs based on workload

5 STATISTICAL RESULTS

In the context of this study, conducting statistical methods like ANOVA (analysis of variance) allows us to assess whether there are statistically significant variations in CPU usage, accuracy, training images, and epochs among different types of client devices used in FL. By exploring these differences, the goal is to identify differences in performance metrics based on device types, which can provide important information about how well FL implementations work across heterogeneous environments.

Table 1 compares CPU performance across different devices, including laptops, MiniPCs, Raspberry Pi 4 (RPI4), and Raspberry Pi 5 (RPI5). The F-value from the one-way ANOVA test indicates significant differences in CPU performance among the devices ($F = 26.428$, $p < 0.01$).

Table 1. Results of one-way ANOVA regarding the differences in CPU by device

Device	N	Mean	Std. Deviation	F	Sig.
Laptop	50	46.528	19.6844	26.428	0.001
MiniPC	50	71.334	15.9116		
RPI4	50	71.224	17.8005		
RPI5	50	74.268	17.6564		

Table 2 presents Tukey's honestly significant difference (HSD) results. Significant differences exist between laptops and other device types. According to these differences, laptops exhibit lower CPU usage compared to the other three types.

Table 2. Results of multiple comparisons regarding the differences in CPU by device

(I) device_id	(J) device_id	Mean Difference (I–J)	Sig.
Laptop	MiniPC	–24.8060*	0.000
	RPI4	–24.6960*	0.000
	RPI5	–27.7400*	0.000

Table 3 presents the results of a one-way ANOVA regarding the differences in accuracy by device type. The observed value of $F = 18.820$, $p < 0.01$ shows that there is a significant difference in accuracy by device type.

Table 3. Results of one-way ANOVA regarding the differences in accuracy by device

Device	Mean	Std. Deviation	F	Sig.
Laptop	0.84032	0.188054	18.820	<0.001
MiniPC	0.68696	0.140836		
RPI4	0.69128	0.141145		
RPI5	0.62826	0.108582		

Table 4 presents the results of the Tukey test regarding the differences in accuracy by device type. Significant differences exist between laptops and other device types. According to these differences, laptops exhibit higher accuracy compared to the other three types. Mini PC is the device that shows the least difference in accuracy compared to laptops, followed by RPI4, and then by the RPI5 device.

Table 4. Results of multiple comparisons regarding the differences in accuracy by device

(I) device_id	(J) device_id	Mean Difference (I–J)	Sig.
Laptop	MiniPC	0.153360*	0.000
	RPI4	0.149040*	0.000
	RPI5	0.212060*	0.000

Table 5 presents the results of a one-way ANOVA regarding the differences in epoch by device types. The observed value of $F = 35.648$, $p < 0.01$, shows that there is a significant difference in epoch by device type.

Table 5. Results of one-way ANOVA regarding the differences in Epoch by device

Device	Mean	Std. Deviation	F	Sig.
Laptop	23.56	14.479	35.648	<0.001
MiniPC	11.36	4.810		
RPI4	10.24	4.128		
RPI5	8.76	3.146		

Table 6 presents the results of the Tukey test regarding the differences in epoch by device type. Significant differences exist between laptops and other device types. According to these differences, the laptop exhibits higher epochs compared to the other three types. Mini PC is the device that shows the least difference in epochs after the laptop, followed by RPI4, and then by RPI5.

Table 6. Results of multiple comparisons regarding the differences in Epoch by device

(I) device_id	(J) device_id	Mean Difference (I–J)	Sig.
Laptop	MiniPC	12.200*	0.000
	RPI4	13.320*	0.000
	RPI5	14.800*	0.000

Table 7 presents the results of a one-way ANOVA regarding the differences in training images by device types. The observed value of F = 2.059, p > 0.05 shows that there is not a significant difference in training images by device type. The laptop has the highest average of training images, followed by the MiniPC and RPI4, while the RPI5 has the lowest average.

Table 7. Results of one-way ANOVA regarding the differences in training images by device

Device	Mean	Std. Deviation	F	Sig.
Laptop	95.44	33.276	2.059	0.107
MiniPC	90.04	31.219		
RPI4	86.08	30.028		
RPI5	80.64	28.584		

5.1 Regression analysis

Table 8 examines the impact of accuracy, epoch, and CPU on the number of training images. The results reveal that all three independent variables—accuracy, epoch, and CPU have a statistically significant impact on the number of training images. Specifically, higher accuracy (β = 214.302, t = 35.069, p < 0.001) and CPU utilization (β = 0.407, t = 15.452, p < 0.001) are associated with increased training images. In summary, accuracy, epoch, and CPU utilization significantly influence the number of training images, with higher accuracy and CPU utilization correlating with more training images.

Table 8. Results of regression analysis

Dep. Variable	*β*	*t*	*p*
Accuracy	214.302	35.069	<0.001
epoch	–1.223	–12.446	<0.001
CPU	0.407	15.452	<0.001

Note: $R = 0.973$, $Adj.R^2 = 0.945$, $F = 1150.364$, $p < 0.001$.

6 DISCUSSION

The AdaptiveMesh algorithm is designed to address challenges in decentralized model training, particularly in heterogeneous environments. By dynamically adjusting training parameters based on client performance metrics, such as CPU utilization and accuracy, the algorithm optimizes resource allocation and improves model convergence. Experimental results demonstrate enhanced training efficiency, with clients adapting workload demands to prevent overheating and optimize training processes. One notable observation from the experiments is the substantial impact of CPU adaptation on training optimization; the algorithm dynamically adjusts computational loads for clients, particularly when CPU utilization exceeds 80%. Regarding model performance, the experimental findings demonstrate notable improvements in accuracy as training progresses. The adaptive FL algorithm facilitates this improvement by dynamically adjusting training parameters based

on client performance metrics, thereby optimizing model convergence and enhancing overall accuracy. The experiments also highlight the dynamic nature of workload allocation, particularly in terms of the number of epochs and training samples assigned to each client. Experiments illustrate the gradual increase of epochs and images across rounds, with clients adapting their workload demands based on CPU utilization and performance metrics. Clients with higher computational capacity consistently request an increase in workload, while those with lower performance dynamically adjust workload demands to prevent overheating and optimize training efficiency.

Further, the statistical results of one-way ANOVA demonstrated significant differences in CPU usage, accuracy, and epochs between laptops and other device types, with laptops showing lower CPU usage, higher accuracy, and a greater number of epochs. The regression analysis revealed that accuracy and CPU significantly impact the number of training images and a number of epochs. Higher accuracy and higher CPU usage correlate with more images and a larger number of epochs. Overall, the experimental results confirm the effectiveness of the adaptive FL algorithm in optimizing model training and resource utilization in heterogeneous environments. The algorithm showed promising results in improving model accuracy and stability across diverse client environments, highlighting its potential for practical deployment in real-world scenarios.

7 CONCLUSIONS AND FUTURE WORK

Federated learning offers a decentralized approach to collaborative model training, ensuring data privacy while harnessing the collective intelligence of distributed clients. This study's contributions include the development of a novel adaptive mechanism for FL, the introduction of a dynamic resource allocation strategy, and a comprehensive evaluation demonstrating significant improvements in efficiency and scalability. *AdaptiveMesh* algorithm enables dynamic adjustments to training parameters, enhancing model performance across heterogeneous wireless environments. This adaptive mechanism guarantees that resource allocation is dynamically adjusted at the clients in order to meet the evolving demands of model training. By providing a detailed overview of the FL system and its adaptive mechanism, this paper contributes to the understanding and advancement of decentralized machine learning. The findings could be relevant and useful in various resource-limited edge environments. The focus is on expanding the implications of these results to embedded IoT devices such as TinyML, where the analyzed design could effectively tackle significant limitations in computing and communication resources. Future work will involve exploring the integration of more advanced machine learning models and larger datasets to assess the scalability and robustness of the algorithm across diverse FL scenarios.

8 REFERENCES

[1] Q. Yang, Y. Liu, T. Chen, and Y. Tong, "Federated machine learning: Concept and applications," *ACM Trans. Intell. Syst. Technol.*, vol. 10, no. 2, pp. 1–19, 2019. https://doi.org/10.1145/3298981

[2] Y. Mao *et al.*, "Communication-efficient federated learning with adaptive quantization," *ACM Trans. Intell. Syst. Technol.*, vol. 13, no. 4, pp. 1–26, 2022. https://doi.org/10.1145/3510587

[3] P. Kairouz *et al.*, "Advances and open problems in federated learning," *arXiv preprint arXiv:1912.04977*, 2021.

[4] Z. Qin, G. Y. Li, and H. Ye, "Federated learning and wireless communications," *IEEE Wireless Communications*, vol. 28, no. 5, pp. 134–140, 2021. https://doi.org/10.1109/MWC.011.2000501

[5] M. J. Sheller *et al.*, "Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data," *Scientific Reports,* vol. 10, 2020. https://doi.org/10.1038/s41598-020-69250-1

[6] T. Li *et al.*, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine,* vol. 37, no. 3, pp. 50–60, 2020. https://doi.org/10.1109/MSP.2020.2975749

[7] L. Shkurti, M. Selimi, and A. Besimi, "FederatedMesh: Collaborative federated learning for medical data sharing in mesh networks," in *Collaborative Computing: Networking, Applications and Worksharing, CollaborateCom 2023*, in Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, H. Gao, X. Wang, and N. Voros, Eds., Springer, Cham., vol. 563, 2024, pp. 154–169. https://doi.org/10.1007/978-3-031-54531-3_9

[8] S. Wang *et al.*, "Adaptive federated learning in resource constrained edge computing systems," *IEEE Journal on Selected Areas in Communications*, vol. 37, no. 6, pp. 1205–1221, 2019.

[9] A. Giuseppi, L. Della Torre, D. Menegatti, and A. Pietrabissa, "Adafed: Performance-based adaptive federated learning," in *Proceedings of the 5th International Conference on Advances in Artificial Intelligence*, 2022, pp. 38–43. https://doi.org/10.1145/3505711.3505717

[10] K. Li, H. Wang, and Q. Zhang, "FedTCR: Communication-efficient federated learning via taming computing resources," *Complex & Intelligent Systems*, vol. 9, pp. 5199–5219, 2023. https://doi.org/10.1007/s40747-023-01006-6

[11] D. Wu, R. Ullah, P. Harvey, P. Kilpatrick, I. Spence, and B. Varghese, "FedAdapt: Adaptive offloading for IoT devices in federated learning," *IEEE Internet of Things Journal*, vol. 9, no. 21, pp. 20889–20901, 2022. https://doi.org/10.1109/JIOT.2022.3176469

[12] R. Mishra, H. P. Gupta, G. Banga, and S. Das, "Fed-RAC: Resource aware clustering for tackling the heterogeneity of participants in federated learning," *IEEE Transactions on Parallel and Distributed Systems*, vol. 35, no. 7, pp. 1207–1220, 2024. https://doi.org/10.1109/TPDS.2024.3379933

[13] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Arcas, "Communication-efficient learning of deep networks from decentralized data," *arXiv preprint arXiv:2102.01936*, 2023.

[14] S. Reddi *et al.*, "Adaptive federated optimization," *arXiv preprint arXiv:2003.00295*, 2021.

[15] Y. Gu, Q. Hu, X. Wang, Z. Zhou, and S. Lu, "FedACS: An efficient federated learning method among multiple medical institutions with adaptive client sampling," in *2021 14th International Congress on Image and Signal Processing, BioMedical Engineering and Informatics (CISP-BMEI)*, 2021, pp. 1–6. https://doi.org/10.1109/CISP-BMEI53629.2021.9624434

[16] P. D. Lorenzo, C. Battiloro, M. Merluzzi, and S. Barbarossa, "Dynamic resource optimization for adaptive federated learning at the wireless network edge," in *ICASSP 2021 – 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 2021, pp. 4910–4914. https://doi.org/10.1109/ICASSP39728.2021.9414832

[17] M. Kundroo and T. Kim, "Efficient federated learning with adaptive client-side hyperparameter optimization," in *2023 IEEE 43rd International Conference on Distributed Computing Systems (ICDCS)*, 2023, pp. 973–974. https://doi.org/10.1109/ICDCS57875.2023.00103

[18] J. Shen, N. Cheng, Z. Yin, Y. Fu, W. Xu, and S. Guo, "RingSFL: An adaptive federated learning system for heterogeneous clients," in *2023 IEEE/CIC International Conference on Communications in China (ICCC)*, 2023, p. 1. https://doi.org/10.1109/ICCC57788.2023.10233324

[19] M. N. Hossen, K. Ahmed, F. M. Bui, and L. Chen, "FedRSMax: An effective aggregation technique for federated learning with medical images," in *2023 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2023, pp. 229–234. https://doi.org/10.1109/CCECE58730.2023.10288859

[20] F. Freitag, P. Vilchez, L. Wei, C.-H. Liu, and M. Selimi, "Performance evaluation of federated learning over wireless mesh networks with low-capacity devices," in *Information Technology and Systems*, Á. Rocha, C. Ferrás, A. Méndez Porras, and E. Jimenez Delgado, Eds., vol. 414, 2022, pp. 635–645. https://doi.org/10.1007/978-3-030-96293-7_53

[21] G. Women and C. M. Center, "Chest x-ray images (pneumonia)," Kaggle. [Online] Available at: https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia. [Accessed: 2021-04-28]

9 AUTHORS

Lamir Shkurti is a Ph.D. candidate at the Faculty of Contemporary Sciences and Technologies, Southeast European University in North Macedonia. He is also employed as a Lecturer at the University for Business and Technology in Kosovo. His research interests include federated learning for resource-constrained platforms, embedded machine learning, Internet of Things, Wireless Mesh Networks, Cloud Computing, virtualization technology, software development and algorithm optimization (E-mail: ls29773@seeu.edu.mk).

Mennan Selimi is an Associate Professor at South East European University, North Macedonia. He is the head of the Distributed Systems and Data Science research group at Max van der Stoel Institute. Previously, he was a Postdoctoral Research Associate at University of Cambridge working with the N4D Lab. He has a Phd in Distributed Computing (with Distinction and Honors) from UPC BarcelonaTech and IST Lisbon. His research interests focus on decentralized cloud infrastructures looking at topics ranging from machine learning and blockchain to network economics. More: https://mvdsi.seeu.edu.mk/mselimi/