FederatedMesh: Collaborative Federated Learning for Medical Data Sharing in Mesh Networks

Lamir Shkurti[1,2], Mennan Selimi[2(✉)], and Adrian Besimi[1]

[1] Faculty of Contemporary Sciences and Technologies, South East European University, Tetovo, North Macedonia
{ls29773,a.besimi}@seeu.edu.mk

[2] Max van der Stoel Institute, South East European University, Tetovo, North Macedonia
m.selimi@seeu.edu.mk

Abstract. Edge computing is a paradigm that involves performing local processing on lightweight devices at the edge of networks to improve response times and reduce bandwidth consumption. While machine learning (ML) models can run on smaller computing devices at the edge, training ML models presents challenges for low-capacity devices. This paper aimed to evaluate the performance of Federated Learning (FL) - a distributed ML framework, when training a medical dataset using Raspberry Pi devices as client nodes. The testing accuracy, CPU usage, RAM memory usage and network performance were measured for different number of clients and epochs. The results showed that increasing the number of devices generally improved the testing accuracy, with the greatest improvement observed in the earlier epochs. However, increasing the number of devices also increased the CPU usage, with a significant increase observed in the later epochs. Additionally, the RAM memory usage increased slightly as the number of clients and epochs increased. The findings suggest that FL can be an effective way to train medical models using distributed devices, but careful consideration must be given to the trade-off between accuracy and computational resources.

Keywords: edge computing · federated learning · mesh networks

1 Introduction

Edge computing complements cloud computing by utilizing local processing on lightweight computing devices, such as IoT gateways and wearable devices, at the edge of networks where data is produced. Local processing on these edge devices can improve response times of cloud services and reduce bandwidth consumption by transmitting less data to the cloud [1,2]. Edge computing is already operational in various industrial and consumer-oriented scenarios and some machine learning (ML) and artificial intelligence tasks can also be moved from the cloud to the edge.

H. Gao et al. (Eds.): CollaborateCom 2023, LNICST 563, pp. 154–169, 2024.
https://doi.org/10.1007/978-3-031-54531-3_9

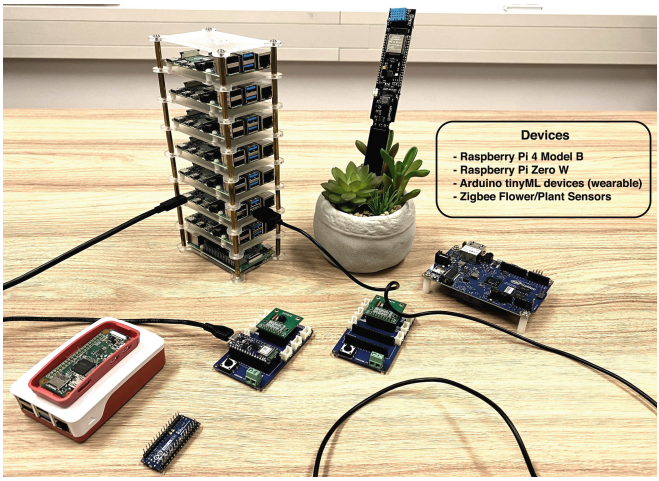

Fig. 1. Resource-constrained devices: Raspberry Pi, Arduino - tinyML and some wearable and IoT devices.

Running ML models on smaller computing devices at the edge is becoming increasingly popular. Even tiny microcontroller boards are now capable of performing simple tasks with trained models [3]. However, training ML models is more computationally demanding and presents challenges for devices with low capacity. While using GPUs (Graphics Processing Units) instead of CPUs (Central Processing Units) can provide better performance, GPUs are not always available on devices like low capacity PCs (mini PCs) or single-board-computers (SBCs). As a result, CPUs have to handle a high load during training, which takes significantly more time compared to high-end devices. This means that training process takes significantly longer on low-capacity devices, making it unsuitable for applications with time constraints.

With the popularization of wearable and mobile devices, intelligent learning applications have been prominently used by many consumers. These devices collect user information about daily activities, providing valuable insights to enhance user lifestyle. The success of smart health applications largely relies on the ability to train ML models on large quantities of user data collected from wearables [4,5]. However, due privacy concerns, security issues, communication overhead, processing delay etc., traditional ML algorithms face challenges that work in a centralized fashion where all the available data is accumulated beforehand. For instance, in the case of wearable systems, privacy remains the key obstacle to implement data analytics algorithms. Most of the time users are sceptical in allowing their personal data to be analysed by the ML algorithms on cloud. Figure 1 depicts several low-constrained devices capable of performing ML functions including Raspberry Pi's, Arduino boards, tinyML devices etc.

Federated Learning (FL) is a paradigm for collaboratively training ML models on computation, storage, energy and bandwidth limited mobile devices in a distributed manner by addressing privacy concerns and reducing communication overhead and processing delay [6,7]. In FL, each node has its own training

data to train a local model, and the subsequent aggregation of the local models leads to a new global model. Wireless mesh networks can be used to support FL on these edge devices, but this can be challenging due to several factors [8]. These challenges include limited resources on low-capacity devices, unreliable network connectivity, device heterogeneity and security and privacy concerns. These challenges can make it difficult to coordinate and aggregate model updates from devices in a FL setting, and to maintain consistent communication between devices [9,9].

The main contributions of the paper is *demonstration of FL in a real resource-constrained wireless mesh environment with dynamic, heterogeneous and intermittent resource availability.* To do this, we assess the possibility of implementing FL on *8* real IoT devices that have limited network and hardware resources and are connected in a wireless mesh network. Therefore, our goal with this contribution is to offer fresh insights that can help make the process of FL more doable on everyday devices that have limited resources. Our aim is to make FL work better on these devices that people commonly use. Further, we present practical observations on the use of resources (CPU, memory, network) and suggestions on the optimal training configurations that must be employed to ensure a satisfactory "training experience" on these low-capacity devices.

The rest of the paper is organized as follows. In Sect. 2 we describe the FL applications used and analyse the state-of-the-art work. Section 3 presents the FL model used. Section 4 provides details about how we set up our experiments, the specific experiments we conducted and the results we achieved. Moving on, Sect. 5 wraps up our findings and conclusions, also suggesting directions for potential future research.

2 Background and Related Work

In this section, initially we provide the background for our work and then present the related work.

2.1 FL Applications and Dataset

Federated Learning (FL) is predominantly utilized in situations that demand a significant emphasis on safeguarding privacy and optimizing resource allocation. The healthcare and medical sector is a major field where FL finds extensive applications. This section of the paper outlines the medical datasets used in the study. The dataset is selected from the healthcare applications mentioned below.

Healthcare Industry: Many hospitals, AI companies and regulatory agencies are responsible for protecting highly sensitive data of the users. In the health industry, many *wearable healthcare* devices are used to monitor patient's health, identify anomalies and treat health conditions. For instance, in each hospital a large amount of real electronic health records (EHR) are needed to train a powerful a medical model. However due to the sensitivity and privacy of medical

data, the demand for a real dataset is hard to be satisfied. FL can solve this by maintaining data anonymity, thus removing many barriers to data sharing.

Dataset: For the FL experiments, we are using the Chest-X-Rays dataset provided in the following link [10]. The Chest-X-Rays dataset consists of 5,863 X-Ray images in JPEG format and 2 categories (Pneumonia/Normal). Chest-X-ray images (anterior-posterior) are selected from retrospective cohorts of pediatric patients of one to five years old from Guangzhou Women and Children's Medical Center, Guangzhou. The FL task to be executed in the experiments is to train a 6-layer Convolutional Neural Network (CNN) model with the Chest-X-Ray dataset. The CNN model has around 420, 000 parameters.

2.2 Related Work

This section describes a review of specific studies that focus on the implementation of FL on devices with limited capacity and research that examines techniques to maintain data privacy, a crucial consideration when training confidential personal health data.

Low-Constrained Devices and Federated Learning: Y. Gao and colleagues [11] conducted a practical assessment of two advanced ML methods - Federated Learning (FL) and split neural networks (SplitNN). The authors looked at different sets of data, tried various types of model designs, involved several users' devices and used different benchmarks to measure how well everything worked. Their study focused on learning, which depended on two types of data: one where the data balance was uneven and another where the data wasn't exactly the same across devices. The training of the model took place on Raspberry Pi devices, and they kept track of how much the CPU and memory was used, the extra communication needed, and how long the training took. Based on their results, FL outperformed SplitNN overall. This is mainly because FL involves less extra communication compared to SplitNN.

We have used some findings from the experiments carried out by Y. Goa on individual Raspberry Pi devices to establish the baseline system in our own research. In particular, we have employed their configuration of 1 and 5 epochs as reference epoch values.

FL for edge environments has been suggested in various studies as surveyed in [12]. Specific types of edge devices are investigated for instance in the Flower framework, where Raspberry Pi, Android phones and NVIDIA Jetson were used [13]. The work on Flower proposes a framework that first addresses the hardware heterogeneity of the clients by providing client-specific software implementations. For instance, the FL client for Android phones consists of a Java implementation applying a specific TensorFlow Lite Model Personalization support for Android Studio. The FL client for the Raspberry Pi and NVIDIA Jetson is implemented in Python.

After looking into similar research, it's clear that various methods have been suggested to make FL use up fewer computer resources. These methods include

adjusting how ML models are trained and even shifting some tasks to other platforms. However, there's still a gap in our knowledge when it comes to how well FL actually works in real wireless mesh setups. Our study focuses on filling this gap. We're taking a hands-on approach by running FL on devices with limited power in mesh networks. This way, we're gathering important information on how to set up FL for different types of user situations.

In [14] study performs a multi-chest disease classification from the CXR images task using the proposed CNN architecture. Also, the authors introduce a new dataset consisting of 28833 CXR images, a mixture of COVID-19, non-covid viral or bacterial pneumonia, lung opacity, and normal cases by aggregating publicly available datasets to apply FL for the proposed models. Through experiments, they compared the results with central training, federated training, and communication-efficient federated training. Also, they have shown that federated training in chest disease classification is achieved with high percent accuracy and is an effective alternative to central training. This study's outcomes can inspire and encourage medical organizations to initiate or adopt their research and practices within the FL approach for chest disease classification.

The authors in the following paper [15] introduces a conceptual framework designed to harness edge computing for healthcare analytics, utilizing user-generated data. The intersection of technologies like cloud computing, edge computing, IoT, wearables, and FL is anticipated to encourage end-users to play a more participatory role in overseeing their health. The main objective is to strengthen patient empowerment and accountability in the domains of monitoring and preventing diseases. This stands as a crucial factor in ensuring the sustainability of contemporary healthcare systems. The proposed model offers the potential for seamless integration of user-generated wellness and behavioral data in an effective and scalable manner.

Privacy Through Federated Learning: Taking privacy as an important aspect while training medical data, the authors are proposing Dopamine [16], a system to train the medical data. This study investigates various ML and privacy-preserving methods. The approach involves using medical data to train Deep Neural Networks - DNNs for medical diagnoses. The training of these DNNs is carried out using distributed datasets through a combination of FL and Differentially-Private Stochastic Gradient Descent - DPSGD.

In a research study mentioned in [17], the authors of the paper combined blockchain technology and FL techniques to train ML models without exposing the actual data. This was particularly relevant for situations where lots of data comes in quickly from connected devices in a setting called the Industrial Internet of Things (IIoT). They pointed out that protecting this data from being leaked in industries is a big deal. To tackle this, they built a secure way of sharing data using blockchain and added privacy-preserving features to FL. They discovered that by using blockchain, they could share the model in the FL process while keeping data private.

Another paper, presented in [18], introduced the BlockFL system, which mixed blockchain with FL. Instead of having a central place (node) that collects

updates from devices, the system used a distributed ledger to exchange these updates between mobile devices. They also studied how quickly learning was completed using this system. In their case, multiple mobile devices were used in order to train the model locally making use of the distributed ledger powered by blockchain technology.

In a different scenario, authors Passerat-Palmbach et al. discussed in [19] how blockchain could help to orchestrate FL for healthcare groups. They showed how data could be contributed to improve ML models while still ensuring privacy and accurate records. They set up a way to track events in the network without revealing who was involved.

Contrary, the work in [20] took a different approach. They proposed using IPFS[1] instead of a single central server for FL. This lets different nodes participate and lead the FL process. They also split the ML models into parts and shared the responsibility for each part among nodes. This was especially useful when dealing with devices that aren't very powerful.

The authors in this work [21] introduce Communication-Efficient Federated Averaging (CE-FedAvg), an adaptation of the Federated Averaging (FedAvg) algorithm FL in the context of IoT devices. The work addresses concerns about data privacy by allowing FL to be performed at the edge servers and gateways without sending data to a central server. CE-FedAvg uses an adapted form of the Adam optimization algorithm and novel compression techniques to reduce the number of communication rounds required for convergence, while also minimizing the data uploaded by clients. Extensive experiments with MNIST and CIFAR-10 datasets, both in IID and non-IID settings, demonstrate that CE-FedAvg achieves faster convergence and requires less total data uploaded compared to FedAvg. Additionally, experiments on a Raspberry Pi testbed confirm that CE-FedAvg can reach a target accuracy in less real time. This research contributes to the field of FL by improving the efficiency and robustness of FL algorithms in edge-computing scenarios.

In the [22] the authors use FL technology for Medical Image Classification to address the issues of medical data security. They present their algorithm FedSLD to utilize knowledge of label distributions to mitigate the challenges posed by data heterogeneity. FedSLD is designed to enhance the training of ML models for medical image classification in a FL setting by leveraging shared label distribution information, ultimately improving the model's accuracy and stability in the presence of data heterogeneity. They used datasets of MNIST, CIFAR10, Organ MNIST, and PathMNIST. This article is important for computer science researchers as it proposes a new method for training ML models in medicine using separated and privatized data. Additionally, the study primarily focuses on model performance without addressing all potential security and privacy challenges related to medical data, which are crucial in healthcare-based ML.

Looking at the reviewed papers in privacy, it's clear that there are various ways to keep data safe in FL. One popular method is called differential privacy and there are also ways to use external tools like blockchain to help. Also,

[1] Interplanetary File System. https://ipfs.io/.

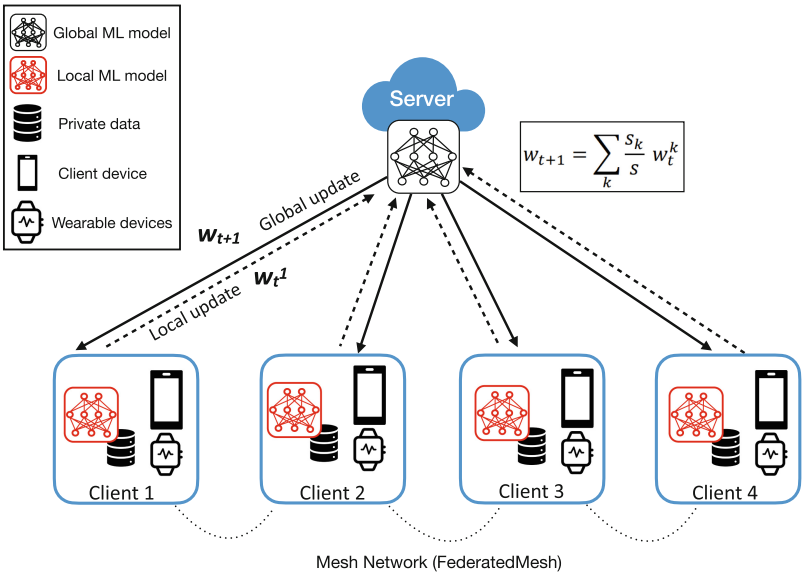

Fig. 2. FL architecture involving communication between the server and clients (such as mobile devices, wearables, etc.)

we noticed that ML applications that use personal information (medical data) and private data (sensor data from the Internet of Things) can gain significant advantages from FL.

3 System Model

3.1 Federated Learning

Federated Learning (FL) has gained significant interest in the research community as a model training technique that enables clients to train models collaboratively without the need to share their local data. FL has become a key area of interest in wearable systems and wireless communications, such as 5G, where edge nodes generate valuable data for applications while still maintaining data privacy [23,24]. FL is a distributed ML technique where numerous clients or workers, such as mobile or wearable devices, train a model in a collaborative manner guided by a central server located in the cloud, as shown in Fig. 2. The training data is stored directly on the devices.

The FL algorithm operates in the following way: the process starts with the server initializing a global model (w_t) and sending this model to all clients. Each of the clients k trains the global model on their own local data for several rounds of training called "epochs". One the local training is complete, the updated model is sent back to the server (w_t^k). The server receives the updated models from all the clients and combines (i.e., merges) them to update the global model (w_{t+1}).

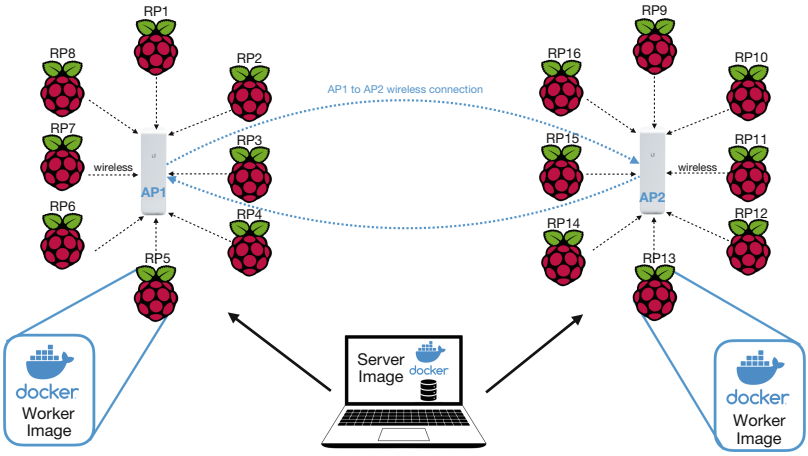

Fig. 3. Testbed used for the experiments

This process is repeated until the model converges or a maximum number of rounds (i.e., iteration of this process) is reached. Client k possesses a training dataset comprising s_k samples as depicted in Fig. 2.

In this section, we focus on a system made of Raspberry Pi edge devices that work together to train ML models using FL. We study *how much CPU and memory* the devices use when training FL models based on healthcare data. Further, we study the *accuracy of the model reached with different number of edge devices* and also *how long does it take to train the models* considering the impact of a real wireless mesh network.

Model: We use FL to train a 6-layer Convolutional Neural Network (CNN) model on the client nodes using the Chest X-Ray Images (Pneumonia) dataset of size 6.7 GB [10,25]. The CNN model has approximately 420,000 parameters.

By sharing our practical experiences, we hope to provide new information that can help improve ML training on edge devices in wireless mesh network.

4 Empirical Analysis and Findings

4.1 Experimental Setup

The setup used for the experiment has actual devices that are connected to each other through a wireless network. The setup consists of 8 Raspberry Pi (RPi) devices that are linked to a pair of wireless access points (4 by 4) as highlighted in Fig. 3. These access points are connected to each other through a wireless link using Ubiquiti Nanostation M5 devices[2]. These devices have a good performance in creating point-to-point links and can operate at a range of up to 15 km at 2.4 GHz and 5 GHz.

[2] https://store.ui.com/collections/operator-airmax-devices/products/nanostation-m5.

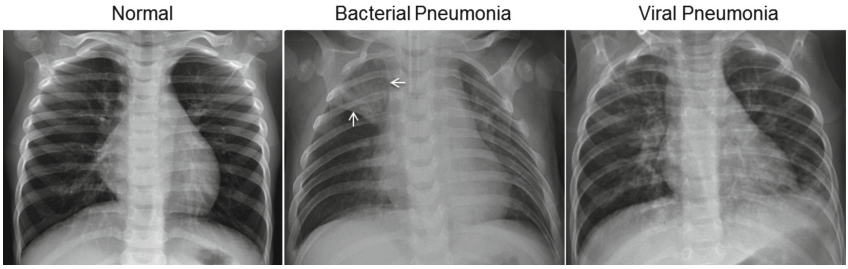

Fig. 4. Examples of chest X-rays in patients with pneumonia as highlighted in [25]

Experiments employ Raspberry Pi 4 Model B devices as computational nodes. The devices feature a robust Quad Core Cortex - A72 processor (1.5GHz), substantial memory (8 GB RAM, 128 GB storage), and an IEEE 802.11ac wireless connection. PyTorch version 1.8.0, OS Raspbian GNU/Linux 10 (buster) and Python version 3.7.3 are used as the software for the experiments. In total, 8 Raspberry Pi devices (4 connected to a single AP) act as workers (clients), and a laptop acts as a server. As highlighted in Fig. 3, as the central node (server) we use a laptop with CPU i5-8250U and 8 GB RAM with Windows 11 Operating System.

The experimental setup consists of two components: the worker and server parts. We developed Docker images for both parts, with the client image installed on Raspberry Pi devices and the server image on the laptop. To train ML models, we utilized the Keras API. During the model training, the data was divided into smaller groups called "batches", each with 10 samples. This allowed the model to be trained in smaller steps, which can improve memory usage and make the training process faster. The training was done for 100 rounds or iterations.

Dataset: Four our experiments the dataset obtained contains 5,863 X-Ray images in jPEG format categorized into two groups (Pneumonia/Normal). The images in this collection are chosen from past groups of young patients, aged one to five years old, at Guangzhou Women's and Children's Medical Center in Guangzhou [10]. Figure 4 highlights some of the examples from the dataset - chest X-rays in patients with different type of pneumonia [25].

4.2 Experimental Results

In the experiments, we aim to measure how accurate the FL model is when it is used with different numbers of clients in mesh networks. First we characterize the network used and also test how much stress the edge devices can handle and measure how much CPU and RAM they use.

Network Characterization. At first, our goal was to assess the performance of the mesh network we are currently utilizing. By utilizing the bandwidth and RTT (Round-trip time) ECDF graph, we can effectively visualize the distribution of our data from the wireless links and identify any irregularities or network issues

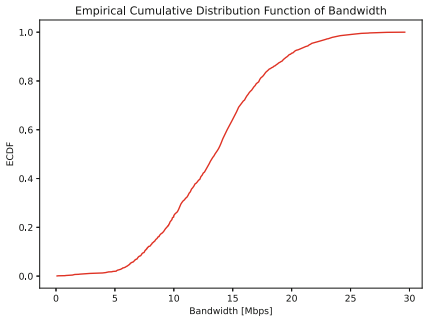

Fig. 5. ECDF - Network Bandwidth

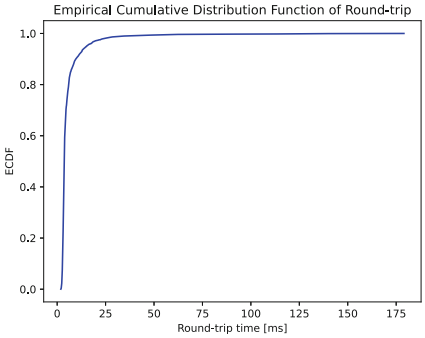

Fig. 6. ECDF - Network RTT

that can make the FL impossible on this network. Figure 5 shows the ECDF (Empirical Cumulative Distribution Function) of the link bandwidth while the FL training experiments happen. The ECDF graph for bandwidth shows that the majority of the links (95%) has a bandwidth between 5 Mbps and 30 Mbps. The bandwidth increases as the ECDF value increases, indicating that higher percentages of the measured data have higher bandwidths. The results suggest that the bandwidth of the data varies widely, with some values as low as 5 Mbps and others as high as 30 Mbps. We can conclude that the *network has sufficient bandwidth to enable the exchange of model updates*, thus FL could be feasible on this network.

Figure 6 shows that the majority of the RTT values are relatively low, with most values falling between 3 ms and 6 ms. This suggests that the network is performing well, with relatively low latency. FL involves the exchange of model updates between client devices and the central server, and lower latency can help to reduce the time required for these exchanges and improve the overall efficiency of the process.

Testing Accuracy: Figure 7 depicts the testing accuracy when 2 and 4 edge devices are used for the training. Figure 7 demonstrates that the testing accuracy improves as the number of epochs increases for both 2 and 4 devices. However, the accuracy is generally higher for 4 devices as compared to 2 devices across all epochs. For the first few epochs (1 and 2, the difference in accuracy between 2 and 4 devices is significant. As the number of epochs increase (4–100), the difference in accuracy between 2 and 4 devices becomes smaller, but 4 devices consistently perform better. The results suggest that using more devices in FL can improve the testing accuracy of the model. However, the improvement *may not be significant after a certain number of epochs*, and other factors such as communication efficiency and device heterogeneity may also impact the performance.

CPU Usage: Figure 8 depicts the average CPU usage when training the model with different number of devices. Figure reveals that the CPU usage varies

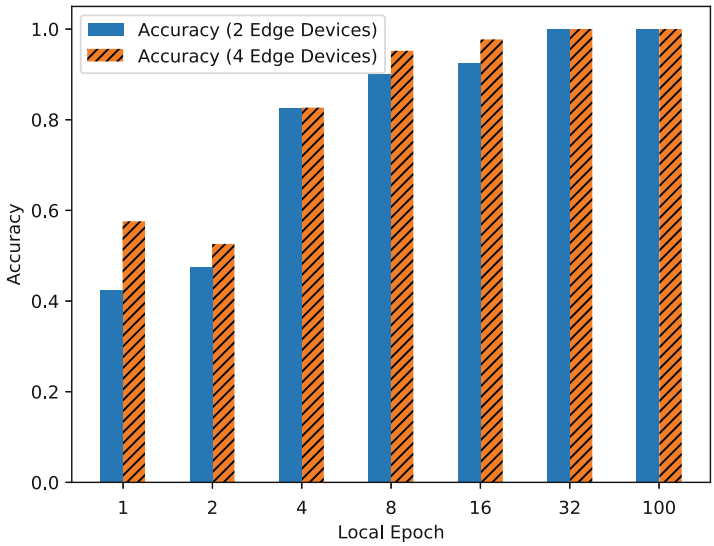

Fig. 7. Testing Accuracy vs Local Epoch (2 and 4 edge devices)

depending on the number of devices used for the training and the number of epoch completed. Additionally, we can see that the CPU usage increased as the number of epochs increased, regardless of the number of devices used. It is also worth noting that the highest CPU usage was observed for the epoch with the highest number of iterations, but even in that case, the CPU usage did not exceed 90% for any of the devices used. Overall, it seems that *training the model with more devices can lead to higher CPU usage*, but using a higher number of devices can also *potentially lead to faster training times*. Further, the peak CPU usage was observed during the model update communication with the client nodes.

Memory Usage: The memory usage during the training of the model with varying numbers of workers is depicted in Fig. 9. Based on the results, we see that the memory usage increased slightly as the number of devices increased for each epoch. However, the difference in memory usage between 2 and 4 devices was relatively small. Thinking about the memory capacity of the devices used in FL is crucial. This is because it might affect how well the training process works and how stable it is overall. In this case, the *Raspberry Pi devices with 8 GB of RAM appeared to handle the memory requirements well*, with relatively stable memory usage across different epochs and number of devices.

The **CPU temperatures** of the 4 Raspberry Pi devices were measured while training FL models. As highlighted in Fig. 10, training FL models requires a lot of computing power, which makes the CPU run at maximum capacity for long periods, reaching on average 80° CPU temperature. Based on the experiments performed, optimizing the training process and reducing the computational workload may help to prevent *overheating*. Overheating in Raspberry Pi

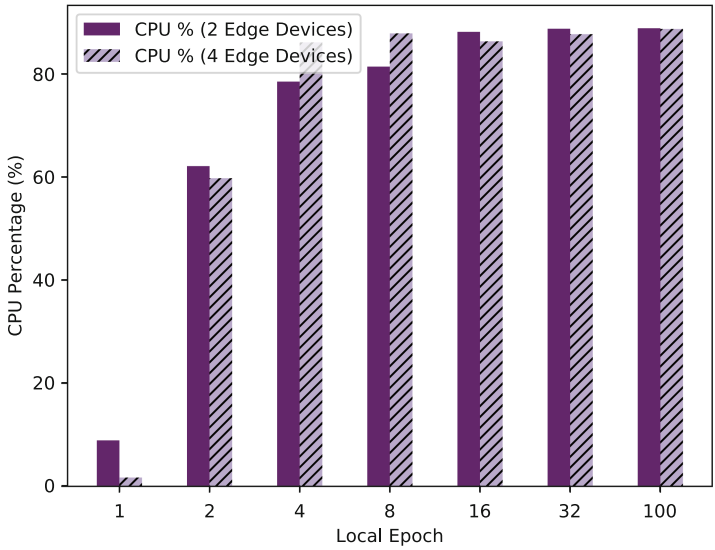

Fig. 8. CPU vs Local Epoch (2 and 4 edge devices)

devices can indeed be a significant concern, as it can lead to the shutdown and interruption of experiments or tasks being performed on the device.

4.3 Discussion

From the experiments performed, we observed that the using the FL in wireless mesh networks can effectively improve the accuracy of the model by increasing the number of devices and epochs. However, this comes at the cost of higher CPU and RAM usage, which needs to be considered when designing a FL system. In this particular study, the use of Raspberry Pi devices with 4 cores of CPU and 8 GB of RAM seems to be sufficient for the task.

It is interesting to note that the CPU usage is higher when communicating with client nodes, which suggests that optimizing the communication protocols can lead to more efficient FL systems. Optimizing the code, using pre-trained models, reducing the training data size or lowering the learning rate can improve the efficiency of the training process. Additionally, the fact that the RAM usage stabilizes after a certain number of epochs suggests that there may be an optimal number of epochs beyond which the model is no longer improving significantly.

Overall, the findings suggest that FL is a promising approach for improving ML models while preserving data privacy, but it requires careful consideration of hardware resources and communication protocols to ensure efficient and effective operation.

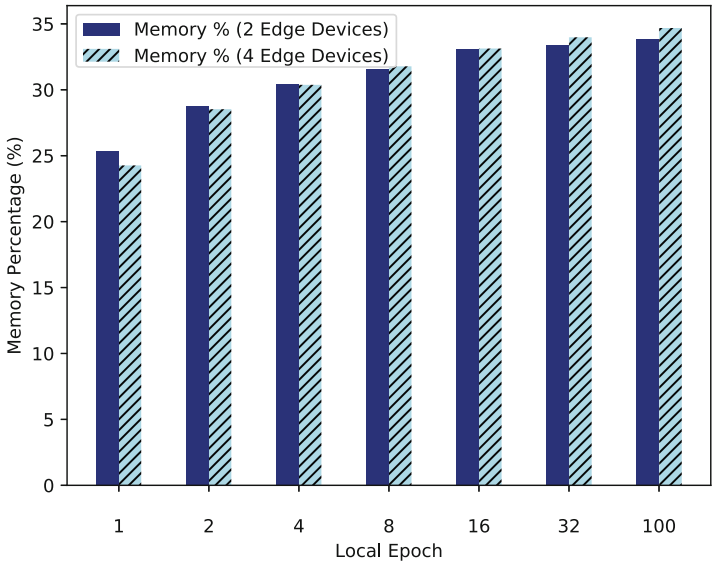

Fig. 9. RAM vs Local Epoch (2 and 4 edge devices)

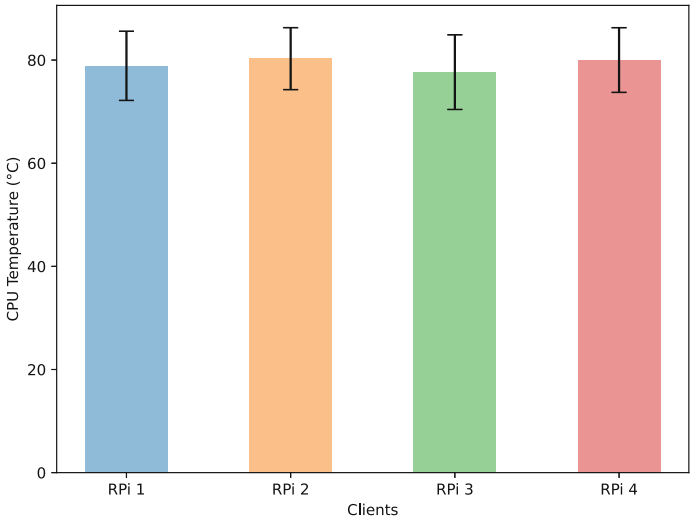

Fig. 10. CPU Temperature (°C) of Raspberry Pi devices

5 Conclusion

Based on the findings and results presented in this study, it can be concluded that FL is a promising technique for training ML models using decentralized data sources. The results showed that increasing the number of devices participating

in the training process led to higher accuracy, especially after a certain number of epochs. However, this came at the cost of increased CPU and RAM usage, which should be considered when designing and implementing FL systems.

One of the major recommendations based on the findings is to carefully choose the number of devices and epochs for training. Finding the right balance between accuracy and how much resources are used is crucial, as training for too long or with too many devices can lead to diminishing returns and increased resource consumption. Additionally, it is recommended to monitor resource usage during training and optimize the system accordingly, for example by using more powerful devices or implementing resource-efficient algorithms.

In conclusion, FL is a powerful technique that can enable training of ML models on decentralized data sources. The findings from this study provide valuable insights into the trade-offs between accuracy and resource usage, and highlight the importance of careful system design and optimization. With further research and development, FL has the potential to revolutionize the field of ML by enabling secure and decentralized training of models on sensitive data. Further, the results obtained suggest that in order to achieve better performance in heterogeneous wireless environments where clients have varying bandwidth and hardware capabilities, it is recommended to develop *FL clients that can dynamically adjust the training parameters*. This will enable a context-aware FL approach, which can better accommodate the diversity of the participating clients and improve the overall efficiency and effectiveness of the training process. Further, the findings may hold potential applicability in various resource-limited edge scenarios. Our goal is to extend these results to embedded IoT devices, where the examined design could help tackle significant constraints in computing and communication resources.

Acknowledgment. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 872614 - SMART4ALL. SMART4ALL is a four-year Innovation Action project funded under call DT-ICT-01-2019: Smart Anything Everywhere - Area 2: Customized low energy computing powering CPS and the IoT. The authors wish to express their gratitude to the SMART4All consortium partners for their valuable comments and feedback, which have contributed to the enhancement of this work.

References

1. Satyanarayanan, M.: The emergence of edge computing. Computer **50**(1), 30–39 (2017)
2. Selimi, M., Lertsinsrubtavee, A., Sathiaseelan, A., Cerdà-Alabern, L., Navarro, L.: Picasso: enabling information-centric multi-tenancy at the edge of community mesh networks. Comput. Netw. **164**, 106897 (2019)
3. Sakr, F., Bellotti, F., Berta, R., De Gloria, A.: Machine learning on mainstream microcontrollers. Sensors **20**(9), 2638 (2020)
4. Arikumar, K.S., et al.: FL-PMI: federated learning-based person movement identification through wearable devices in smart healthcare systems. Sensors **22**(4), 1377 (2022)

5. Farhad, A., Woolley, S., Andras, P.: Federated learning for AI to improve patient care using wearable and IoMT sensors. In: 2021 IEEE 9th International Conference on Healthcare Informatics (ICHI), p. 434 (2021)
6. Yang, Q., Liu, Y., Chen, T., Tong, Y.: Federated machine learning: concept and applications. ACM Trans. Intell. Syst. Technol. **10**(2), 1–19 (2019)
7. Yang, K., Jiang, T., Shi, Y., Ding, Z.: Federated learning via over-the-air computation. IEEE Trans. Wireless Commun. **19**(3), 2022–2035 (2020)
8. Pinyoanuntapong, P., Janakaraj, P., Wang, P., Lee, M., Chen, C.: Fedair: towards multi-hop federated learning over-the-air. In: 2020 IEEE 21st International Workshop on Signal Processing Advances in Wireless Communications (SPAWC), pp. 1–5 (2020)
9. Freitag, F., Vilchez, P., Wei, L., Liu, C.H., Selimi, M.: Performance evaluation of federated learning over wireless mesh networks with low-capacity devices. In: Rocha, Á., Ferrás, C., Méndez Porras, A., Jimenez Delgado, E. (eds.) ICITS 2022, pp. 635–645. Springer, Cham (2022). https://doi.org/10.1007/978-3-030-96293-7_53
10. Women, G., Center, C.M.: Chest X-ray images (pneumonia). https://www.kaggle.com/datasets/paultimothymooney/chest-xray-pneumonia. Accessed 28 Apr 2021
11. Gao, Y., et al.: End-to-end evaluation of federated learning and split learning for internet of things. In: 2020 International Symposium on Reliable Distributed Systems (SRDS), pp. 91–100 (2020)
12. Abreha, H.G., Hayajneh, M., Serhani, M.A.: Federated learning in edge computing: a systematic survey. Sensors **22**(2), 450 (2022)
13. Mathur, A., et al.: On-device federated learning with flower (2021)
14. Cetinkaya, A.E., Akin, M., Sagiroglu, S.: A communication efficient federated learning approach to multi chest diseases classification. In: 2021 6th International Conference on Computer Science and Engineering (UBMK), pp. 429–434 (2021)
15. Hakak, S., Ray, S., Khan, W.Z., Scheme, E.: A framework for edge-assisted healthcare data analytics using federated learning. In: 2020 IEEE International Conference on Big Data (Big Data), pp. 3423–3427 (2020)
16. Malekzadeh, M., Hasircioglu, B., Mital, N., Katarya, K., Ozfatura, M.E., Gunduz, D.: Dopamine: differentially private federated learning on medical data. arXiv abs/2101.11693 (2021)
17. Lu, Y., Huang, X., Dai, Y., Maharjan, S., Zhang, Y.: Blockchain and federated learning for privacy-preserved data sharing in industrial IoT. IEEE Trans. Industr. Inf. **16**(6), 4177–4186 (2020)
18. Kim, H., Park, J., Bennis, M., Kim, S.: On-device federated learning via blockchain and its latency analysis. CoRR abs/1808.03949 (2018)
19. Passerat-Palmbach, J., Farnan, T., Miller, R., Gross, M.S., Flannery, H.L., Gleim, B.: A blockchain-orchestrated federated learning architecture for healthcare consortia. CoRR abs/1910.12603 (2019)
20. Pappas, C., Chatzopoulos, D., Lalis, S., Vavalis, M.: IPLS: a framework for decentralized federated learning (2021)
21. Mills, J., Hu, J., Min, G.: Communication-efficient federated learning for wireless edge intelligence in IoT. IEEE Internet Things J. **7**(7), 5986–5994 (2020)
22. Luo, J., Wu, S.: FedSLD: federated learning with shared label distribution for medical image classification. In: 2022 IEEE 19th International Symposium on Biomedical Imaging (ISBI), pp. 1–5 (2022)
23. Niknam, S., Dhillon, H.S., Reed, J.H.: Federated learning for wireless communications: motivation, opportunities, and challenges. IEEE Commun. Mag. **58**(6), 46–51 (2020)

24. Ibraimi, L., Selimi, M., Freitag, F.: Bepoch: improving federated learning performance in resource-constrained computing devices. In: IEEE Global Communications Conference (GLOBECOM) (2021)
25. Kermany, D.S., et al.: Identifying medical diagnoses and treatable diseases by image-based deep learning. Cell **172**(5), 1122–1131.e9 (2018)