future internet

Article

Implementation and Evaluation of a Federated Learning Framework on Raspberry PI Platforms for IoT 6G Applications

Lorenzo Ridolfi, David Naseh, Swapnil Sadashiv Shinde and Daniele Tarchi *

Department of Electrical, Electronic and Information Engineering "Guglielmo Marconi", University of Bologna, 40126 Bologna, Italy; lorenzo.ridolfi6@studio.unibo.it (L.R.); david.naseh2@unibo.it (D.N.); swapnil.shinde2@unibo.it (S.S.S.)

* Correspondence: daniele.tarchi@unibo.it

Abstract: With the advent of 6G technology, the proliferation of interconnected devices necessitates a robust, fully connected intelligence network. Federated Learning (FL) stands as a key distributed learning technique, showing promise in recent advancements. However, the integration of novel Internet of Things (IoT) applications and virtualization technologies has introduced diverse and heterogeneous devices into wireless networks. This diversity encompasses variations in computation, communication, storage resources, training data, and communication modes among connected nodes. In this context, our study presents a pivotal contribution by analyzing and implementing FL processes tailored for 6G standards. Our work defines a practical FL platform, employing Raspberry Pi devices and virtual machines as client nodes, with a Windows PC serving as a parameter server. We tackle the image classification challenge, implementing the FL model via PyTorch, augmented by the specialized FL library, Flower. Notably, our analysis delves into the impact of computational resources, data availability, and heating issues across heterogeneous device sets. Additionally, we address knowledge transfer and employ pre-trained networks in our FL performance evaluation. This research underscores the indispensable role of artificial intelligence in IoT scenarios within the 6G landscape, providing a comprehensive framework for FL implementation across diverse and heterogeneous devices.

Keywords: federated learning; transfer learning; virtual machines; raspberry PI; proof-of-concept

Citation: Ridolfi, L.; Naseh, D.; Shinde, S.S.; Tarchi, D. Implementation and Evaluation of a Federated Learning Framework on Raspberry PI Platforms for IoT 6G Applications. *Future Internet* **2023**, *15*, 358. https://doi.org/10.3390/fi15110358

Academic Editors: Qiang Duan and Zhihui Lu

Received: 28 September 2023
Revised: 27 October 2023
Accepted: 30 October 2023
Published: 31 October 2023

1. Introduction

The upcoming 6G technology is expected to create an intelligent, fully connected, and digitized society through large-scale deployments of distributed intelligent networks [1]. This is expected to create a plethora of new intelligent services and applications with specific demands. Internet of Things (IoT) technology has foreseen great success through enabling 5G solutions and is expected to play a key role in 6G society [2]. With the integration of IoT subcases into different wireless scenarios, many sensory nodes are expected to be deployed in the 6G world, with the ability to sense nearby environments and produce tons of high-quality data. This data can be extremely important to enable intelligent solutions in the 6G networks [3].

Machine Learning (ML) is another important technology that has acquired a central role in the 6G vision; in particular, to enable intelligent services [4,5]. Various ML methods are expected to be deployed in different wireless scenarios to enable intelligent solutions. ML can be extremely useful for analyzing the 6G network data and harnessing its intelligence. The Centralized Learning (CL) method is one of the most common approaches to generating AI algorithms in distributed communication scenarios. This involves concentrating all training data on a single server, which can complete the model's training individually due to its high computing power. However, this solution may not be particularly efficient when analyzed from a 6G perspective. Next-generation networks will need

to handle large amounts of information, most of which will be produced by devices at the network's edge. In this scenario, moving data to a central server is expensive in terms of communication resources and represents a potential risk of network overload. It can induce large communication overheads and has limited applicability in the resource-constrained and time-critical 6G world. Furthermore, privacy and security restrictions can also limit the possibility of data transmissions to server nodes.

A more efficient alternative is offered by Distributed Learning (DL), which is based on decentralization of training [5]. In this approach, the training phase will be carried out directly on end devices or possibly on specific nodes located strategically on the edge of the network. DL allows for the training of models by processing the local data of each device directly within it. This approach enables the use of large amounts of heterogeneous data scattered throughout the vast network, involving a larger audience of users than the centralized variant. This approach also introduces additional advantages. First, it makes it possible to significantly reduce data flow, communication overhead, and network traffic. Second, there is greater protection for end-user privacy as data are processed directly on personal terminals. Finally, by distributing the computational load among the multitude of devices, it is also possible to obtain essential improvements in terms of energy management. Several forms of distributed learning, such as Federated Learning (FL), collaborative learning, split learning, and multi-agent reinforcement learning, are widely considered in wireless networks for enabling intelligence-at-the-edge. Among others, FL has achieved great success in building high-quality ML models based on dispersed wireless data [6,7]. FL is also a candidate for next-generation 6G communication standard allowing for setting up an intelligence-at-the-edge framework [8].

The traditional FL approach includes a set of devices with datasets and a server node [6]. The server node initiates the learning process by defining the learning task, and corresponding model, which is then transmitted to FL devices. With the help of received model parameters from the server node (also called the global model), their datasets, and onboard computation capabilities, each FL device is expected to train the ML model locally. Then, the updates to the ML model from each device are sent to the server. After receiving the updates from the devices, the server node can use the aggregation function (e.g., Federated Averaging (FedAvg)) to create a new global model. This aggregation phase allows devices to share their training knowledge. The devices then use the new global model in the next round of model local training. The process lasts for several rounds till some predefined stopping criteria are fulfilled. Though this approach has several advantages in terms of reduced data transmission costs, enhanced data privacy, etc., and several new challenges have emerged. One of the major challenges in the FL framework is the presence of heterogeneous nodes with different capabilities, in terms of available datasets, computation power, etc. The presence of heterogeneous nodes can be common in different 6G scenarios. Also, the amount of time required to achieve the convergence of the FL model can be unaccepted in latency-critical 6G use cases. Therefore, it is important to analyze the performance of the traditional FL approach in the presence of heterogeneous devices to have a common understanding of their behaviors and possible solutions to tackle the challenges.

1.1. Technological Background

ML technology has gained a lot of attention for enabling intelligent solutions in wireless networks including mobile communication networks [9], wireless sensor networks [10], transportation systems [6], non-terrestrial networks [11]. Complex wireless communication problems such as resource management [12], data offloading toward edge networks [13], spectrum management [14], routing [15], user-server allocation [7], etc., can effectively solve through different ML techniques. Among others, FL is widely considered a distributed learning approach to provide efficient learning solutions. In one paper [16], the authors proposed energy-efficient FL solutions in wireless communication environments. Work in [17] discusses the applicability of FL solutions in smart city environments. FL is also widely

used to solve vehicular network problems, especially in edge computing environments [6]. FL solutions are also considered over different satellite networks [18]. However, these works have analyzed the performance of FL solutions in different wireless environments, without considering the practical implementations.

Recently, several authors considered a learning testbed implemented through different sensory nodes to measure the ML solution's performances. Raspberry Pi devices are commonly considered to analyze the performance of ML solutions proposed to solve different wireless networking problems. In [19], the authors proposed Reinforcement Learning (RL)-based solutions for efficient routing processes in software-defined wireless sensor networks. The testbed includes Raspberry Pi devices as sensor nodes to analyze the performance of RL solutions. In one paper [20], the proposed deep learning-based solutions for detecting a speed bump in an intelligent vehicular system to assist drivers are tested with the help of Raspberry Pi devices and associated camera modules. In one paper [21], an approach called Distributed Incremental Learning (DIL) was introduced to mitigate "Catastrophic Forgetting" in healthcare monitoring. However, the large model size (49 KB) poses challenges for Raspberry devices, and there is a lack of data batch quality details affecting convergence. In one paper [22], the proposed communication-efficient FL solutions are tested with the help of Raspberry Pi devices in edge computing environments. However, to the best of our knowledge, the most recent literature lacks the study associated with the analysis of FL performance in the presence of heterogeneous clients. Given the importance of different IoT subsystems with heterogeneous nodes, such studies can be extremely useful from the 6G network's perspectives. This is one of the main motivations behind this experimental analysis of the FL process in the presence of diverse sets of clients.

1.2. Contributions and Novelties

In this study, we present an in-depth exploration of Federated Learning (FL) methodology within the context of 6G technology, delving into its intricacies and challenges across heterogeneous nodes. Our work uniquely integrates hardware and software components, utilizing a distinctive combination of Raspberry Pi devices and virtual machines as FL clients, each equipped with diverse datasets sourced from the CIFAR10 dataset—a widely acknowledged benchmark in image classification. Our contributions and innovations can be outlined as follows:

- **Cooling Mechanism Impact (Section 4.1):** We meticulously investigate the influence of cooling mechanisms on training accuracy, underscoring their practical significance in accelerating model convergence, especially in resource-constrained environments. This detailed analysis, expounded on in Section 4.1, elucidates the pivotal role of cooling mechanisms, providing valuable insights into optimizing FL performance.
- **Heterogeneous Client Compensation (Section 4.2):** Through a thorough exploration of asymmetric data distribution scenarios, both with and without random selection, we dissect the intricate dynamics of FL performance. Our study highlights the delicate balance necessary in distributing training data among diverse nodes, revealing the complexities of FL dynamics in real-world scenarios. These findings, presented in Section 4.2, offer critical insights into the challenges and solutions concerning data heterogeneity in FL setups.
- **Overfitting Mitigation Strategies (Section 4.2):** We tackle the challenge of overfitting in FL by implementing meticulous strategies. By integrating random selection techniques, we effectively mitigate overfitting risks, optimizing model generalization and ensuring the resilience of FL outcomes. This contribution, outlined in Section 4.2, underscores our commitment to enhancing the robustness of FL models.
- **Scalability Analysis (Section 4.3):** Our study provides a comprehensive exploration of FL scalability, assessing its performance with an increasing number of users. This analysis, detailed in Section 4.3, offers crucial insights into FL's scalability potential, essential for its integration in large-scale, dynamic environments. It emphasizes the

system's adaptability to diverse user configurations, laying the foundation for FL's applicability in real-world scenarios.

- **Pretraining Effectiveness (Section 4.4):** We delve into the effectiveness of pretraining techniques in enhancing accuracy rates. Pretraining emerges as a potent tool, significantly boosting the model's performance and showcasing its potential in optimizing FL outcomes. This contribution, discussed in Section 4.4, highlights the practical implications of pretraining in FL applications, providing actionable insights for future implementations.
- **Transfer Learning Impact (Section 4.5):** In Section 4.5, we investigate the potential of Transfer Learning, evaluating its impact under diverse client configurations. Our results,underscore Transfer Learning's capacity to enhance FL model performance, especially in the face of varied client scenarios. This analysis showcases Transfer Learning's adaptability in real-world applications, emphasizing its role in improving FL outcomes across dynamic and heterogeneous environments.

These contributions collectively form a comprehensive and innovative exploration of FL dynamics, addressing key challenges and offering practical solutions essential for the advancement of Federated Learning technologies in complex, real-world settings.

1.3. Limitations

Despite these contributions, our study acknowledges certain limitations. While our findings offer valuable insights, the scope of this research is confined to specific FL configurations and dataset characteristics. Further exploration of different FL architectures, diverse datasets, and real-world deployment challenges remains an area ripe for future investigation. Additionally, the scalability analysis, while comprehensive, focuses on a limited range of users and could benefit from further exploration with more extensive user groups in practical scenarios.

2. Distributed Machine Learning

The traditional ML approach was based on the centralized structure, where distributed wireless nodes, a potential data source, were needed to transmit their data samples to the centralized, more powerful node with a large amount of storage and computation power. With growing interest in the 5G system and the upcoming 6G technology, the wireless world is filled with tiny devices capable of sensing the environment and generating tons of high-quality data. Such data can be extremely large, and if a traditional centralized approach is adopted, it can induce a huge communication overhead. On the other hand, with the presence of such a large number of devices, the global dataset generated at the centralized server node (i.e., through the accumulation of data from the distributed nodes) can be extremely large, inducing much higher training costs. In addition to this, novel intelligent services and applications are based upon stringent requirements in terms of latency, privacy, reliability, etc. This presents challenges when considering centralized ML model training for wireless scenarios. However, with recent innovations in hardware/software domains, the end devices' onboard capabilities have increased by several folds. With this new capability, these devices can train fairly complex ML models locally with their own datasets. This can omit the requirements for long-distance data communication and additional training overhead. In addition to this, devices can communicate with each other and server nodes to fine-tune the ML models with improved performances. This has opened a new trend of ML model training called distributed learning, for countering the drawbacks of traditional centralized methods. There are various forms of distributed learning methods considered in the recent past.

There are two main approaches available for performing distributed learning: with data in parallel or with models in parallel [23]. The former involves distributing training data on several servers, while the latter divides the model's parameters between different servers. However, implementing the parallel model approach is difficult due to the complexity of dividing machine learning models into distinct groups of parame-

ters. Therefore, most distributed machine learning implementations work through data distribution. Federated Learning (FL), collaborative learning, Multi-agent Reinforcement Learning (MARL), and split learning are some of the most important distributed learning methods. Among others, FL has been widely used in wireless networks to enable intelligent solutions efficiently.

FL is a framework for distributed machine learning that protects privacy by operating through decentralized learning directly on end devices. It involves a certain number of clients, distributed throughout the network, each of whom trains their local model using the data at their disposal. After training, the clients send their models to a central server, which aggregates them into a single neural network and then transmits it to the end devices. This is an iterative process, with each iteration called a federated round. The objective of federated learning is to minimize a function and ensure efficient FL, several variables must be considered [24].

For the efficient implementation of FL, it is imperative to take into account several variables, encompassing the following:

- Selection of devices for learning
- Disparities in performance levels among the clients in use
- Management of heterogeneous training data
- Potential algorithms for local models' aggregation
- Selection of a proper aggregation Strategy at the Parameter Server
- Resource allocation

Concerning the choice of devices, specific parameters, including the quality and quantity of local data, connection performance with the central server, and computational performance of the client, need to be evaluated.

In a heterogeneous environment that involves wireless nodes with various onboard capabilities, it is crucial to pay attention to differences in the computing capabilities of the devices. In fact, a client with reduced computational capabilities will require a longer time to train the model locally, thereby risking the deceleration of the entire learning process. The following two FL approaches are widely considered to enable distributed learning [25] and can be impacted by the heterogeneous nature of the computational capabilities of the devices:

- **Synchronous FL:** All devices participate in training the local models for a specific period, sending the parameters to the central server. In this case, the server receives the client models simultaneously and aggregates them with the certainty that it is using the contribution of all the devices. However, this approach poses some challenges, in the case of heterogeneous client nodes having different capabilities. In such cases, the less-performing clients are compelled to invest more resources to complete the training within the expected timeline. To match the latency performance of other, high-performing clients with more resources, devices can only use a subset of their data.
- **Asynchronous FL:** In this case, there are no time restrictions for local training operations, with each device training its model based on its own capabilities, after which it sends the parameters to the server that proceeds with aggregation. This approach is more appropriate even in the presence of unstable network connections, where a device without network access can continue to train its model until it reconnects. Such an asynchronous approach can potentially reduce the number of FL devices participating in the individual FL rounds. This also requires more complex server-side operations to manage the devices according to their needs.

In Section 4, we explore these FL approaches when evaluating the system performance; more specifically, we employ asynchronous FL in heterogeneous client compensation (Section 4.2), to tackle the challenge posed by the discrepancy between the relatively sluggish Raspberry PIs and the swift Virtual Machines, which ultimately leads to a decline in the overall accuracy of the aggregated data. This measure shall be taken to restore the balance between the two.

In another case, the presence of heterogeneous amounts of data on FL devices can also largely impact FL performance. In one paper [26], the authors propose federated continual learning to improve the performance of Non-IID data by introducing the knowledge of the other local models; however, the paper does not address the scalability of the proposed method for large-scale distributed learning systems. In such scenarios, locally generated models at different FL nodes may have distinct characteristics from each other and may not lead to proper convergence when aggregated into a single neural network. The convergence problem can directly impact the algorithm chosen for the aggregation. In particular, the traditional federated average (FedAvg) approach can have limited performance since it does not distinguish the model parameters from different devices (i.e., simple averaging) [25]. In another case, proper weights can be assigned to each device's models according to their quality (i.e., weighted average). In some cases, device selection policies can be adapted to avoid the participation of FL devices with imperfect models.

The use cases mentioned above indicate the importance of defining a proper FL framework in heterogeneous environments based on the properties of the FL device and the characteristics of the environment. A one-fit-all FL approach can have reduced performance in different cases. Therefore, it is vital to analyze the performance of FL models in various scenarios and to select a proper FL model.

3. Implementation of the System

The primary goal of this work is to design and analyze a comprehensive and well-structured FL framework to train machine learning algorithms using a federated approach involving heterogeneous FL clients. Here, we outline the important steps considered during the implementation of the FL process.

The considered FL system is based on a client-server architecture, with the server represented by a Windows PC and a set of Raspberry Pi devices acting as FL clients. In addition, a set of virtual clients are also considered, to build an FL framework with heterogeneous clients. Given the importance of network programmability and virtualization technologies in the 5G/6G networks, defining the FL framework with a set of hardware/software clients can have added advantages. Finally, inter-device communication is carried out through the local network, to which all hardware nodes are connected via Wi-Fi.

As mentioned above, the main objective is to evaluate the efficacy of FL in the context of devices with limited and heterogeneous resources.

We have considered a typical image classification problem and aim to build a proper Deep Neural Network (DNN) with the help of the considered FL framework. The well-known CIFAR10 dataset is considered during model training operations. It should be noted that to induce data heterogeneity over different FL clients, the original dataset is split into different datasets. Also, though the analysis is performed with a specific ML task with predefined datasets, this can be extended to any generic ML problem. The FL framework is built with the help of Python programming language. In particular, the PyTorch library is considered to train the neural network model, while Flower, a specialized library for federated machine learning, is considered to automate the client-server interactions more efficiently.

In the following, we describe in detail the various configurations used in client/server parts of the considered FL model. Figure 1, presents the basic elements of the considered FL framework that includes a set of Raspberry Pi devices, virtual machines, and an FL server.

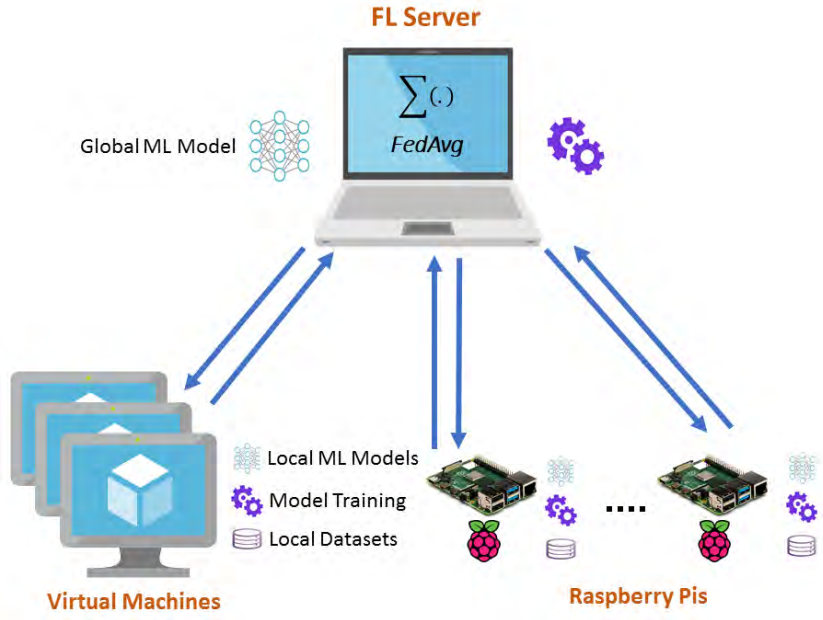

Figure 1. Considered FL Framework with Heterogeneous Clients.

3.1. Server Configurations and Functionalities

Here we introduce the main steps required to configure the server part of the FL system and the corresponding software employed. The FL server is installed on a Windows computer, which exhibits high-performance levels compared to the client devices. Specifically, the server is an Asus ROG Zephyrus S GX502GV with an Intel Core i7-9750H 6 x 2.6–4.5 GHz, Coffee Lake-H processor, NVIDIA GeForce RTX 2060 Mobile (6 GB VRAM, GDDR6 graphics card) and 16 GB DDR4–2666 Hz RAM. Furthermore, Wi-Fi connectivity is 802.11 a/b/g/n/ac with Intel Wireless-AC 9560.

We used the Anaconda platform, which allows the development of Python-based solutions with advanced package and library management. The current project was developed using the basic Anaconda environment, which includes Python 3.9.13. We downloaded and installed two further libraries, namely PyTorch version 2.0.0 and Flower version 1.3, through the integrated terminal. The framework is built around three Python scripts: one to run the server, another to run the client, and a third to define specific neural network training methods. It should be noted that the system can only communicate on the local network. Data are transmitted and received between the various devices using specific methods defined in the scripts.

The FL server needs to perform several functions to enable the FL. At first, the server establishes communication with a considered set of clients through the local network. Next, it initializes the global DNN model. DNN model can be initialized through random parameters or pre-trained models saved in memory based on the considered scenarios. After that, the network parameters are transmitted to all clients to have a common starting point for federated training.

Throughout the federated training process, clients train their models locally for a specified number of local rounds; then, they send the respective local model parameters to the server. The server then aggregates the parameters of all the models received from clients into a single neural network. For this implementation, the chosen algorithm for aggregating the parameters of all models received from clients into a single neural network is the federated average (FedAvg). FedAvg is an FL algorithm that enables collaborative model training across multiple devices or clients while preserving data privacy. It aggregates local model updates from individual clients to create a global model. Its procedure and formula are described in Algorithm 1.

Algorithm 1: Federated Averaging (FedAvg) Algorithm

Input: Global model parameters θ
Output: Updated global model parameters θ
Initialization: Initialize θ with random values;
while *not converged* **do**
  for *each client i in the federated network* **do**
    Compute local model update w_i using client's local data;
    Send w_i to the server;
  end
  Aggregation (FedAvg):
    Compute updated global model parameters θ as the average of received local updates;
  $\theta = \frac{1}{N} \sum_{i=1}^{N} w_i$;
end

The performance of the aggregated model is then assessed in terms of accuracy and losses using a pre-loaded test data set on the server. The results of this assessment are then recorded in a CSV file. After that, the new aggregated model is returned to the clients to repeat the training procedure with the next federated round. The described process is reiterated until the designated number of iterations is reached. Each of these functions is important for enabling the FL process.

3.2. Client Configurations and Functionalities

Client devices include a set of Raspberry Pi devices along with the virtualized clients in the form of virtual machines. In particular, Raspberry Pi 3B+ is considered as a client during the experiments. The Raspberry Pi 3B+ is equipped with a Broadcom BCM2837B0 processor, Cortex-A53 (ARMv8) 64-bit SoC with a clock speed of 1.4 GHz, a 1GB LPDDR2 SDRAM, and 2.4 GHz and 5 GHz IEEE 802.11.b/g/n/ac wireless LAN. Moreover, it has Bluetooth 4.2, BLE, Gigabit Ethernet over USB 2.0 (max throughput 300 Mbps), a 40-pin GPIO header, Full-size HDMI, 4 x USB 2.0, DSI, a Micro SD slot, and a power supply of 5V/2.5A DC.

To facilitate the experiment, the Raspberry PI OS 64-bit operating system was installed on each Raspberries through the official imager. The 64-bit system is necessary for the proper functioning of PyTorch, which currently does not support 32-bit variants. Configuring the devices via SSH enabled the Virtual Network Computing (VNC) service, which uses the local network to transmit the Raspberry desktop to the connected Windows computer. The remote connection via the cloud could further extend the functionality of VNC. Such a procedure allowed interaction with the Raspberry desktop directly on the notebook that hosts the server, facilitating monitoring of the simulation's progress from a single screen.

The client-side code was executed in the Python environment pre-installed on the Raspberry Pis and updated to version 3.9.2. Moreover, the installation of the PyTorch and Flower libraries was necessary. The clients are based on the client.py script, which inherits all the contents of the file cifar.py. This file is essential for generating, training, and evaluating the neural network. Such a configuration allowed the devices to participate in the training process by sharing their computational power with the server. In this way, training time and computational costs can be reduced. The Raspberry Pi 3B+ devices were a suitable choice for this experiment due to their low cost and high customizability. They allow users to modify the hardware and software configurations, facilitating the implementation of ML models and algorithms. Raspberry Pi devices can also be used in various applications, such as robotics, automation, and the Internet of Things (IoT). The proposed setup could be replicated in various scenarios where training machine learning models on devices with limited resources could be beneficial.

In addition to the Raspberry devices, up to eight virtual clients are also considered during the experimentation. It should be noted that all the virtual clients were installed on the same PC. However, this can be easily extended to multi-PC scenarios to enable more diverse sets of clients. Such an approach is beyond the scope of this work. In general, virtual clients have more resources compared to hardware clients. The virtual clients are based on the same scripts used in the implementation of the physical clients, and their execution simultaneously is facilitated using multiple Python terminals open on different screens. A cooling fan was necessary to prevent excessive degradation of computational performance in the Raspberry devices due to overheating. The fan was operational during all the tests conducted to maintain the optimal performance of the devices, as indicated in Figure 2. Later, in the simulation and performance evaluation section, we explore the effect of cooling on accuracy and convergence rate.

Figure 2. Cooling Mechanism for Raspberry Pi Devices.

In the FL framework, client nodes are required to perform a distinctive set of functions. In the beginning, each client should establish a communication channel with the server through a local network. At the beginning of each FL round, the client should receive the updated global model parameters from the server machine. These parameters along with the local data should be used to update the ML model through local training operations. Client devices should train the neural network on local data, iterating for a certain number of periods, i.e., local epochs. Furthermore, it is important to evaluate the performance of the newly trained local model using a local test data set and send the obtained metrics to the server. Finally, the client should transmit the parameters of the trained model back to the server. Figure 3, provides a detailed experimental setup used during the implementation of FL with the help of Raspberry Pi nodes, a virtual machine, and an FL server installed on a Windows PC.

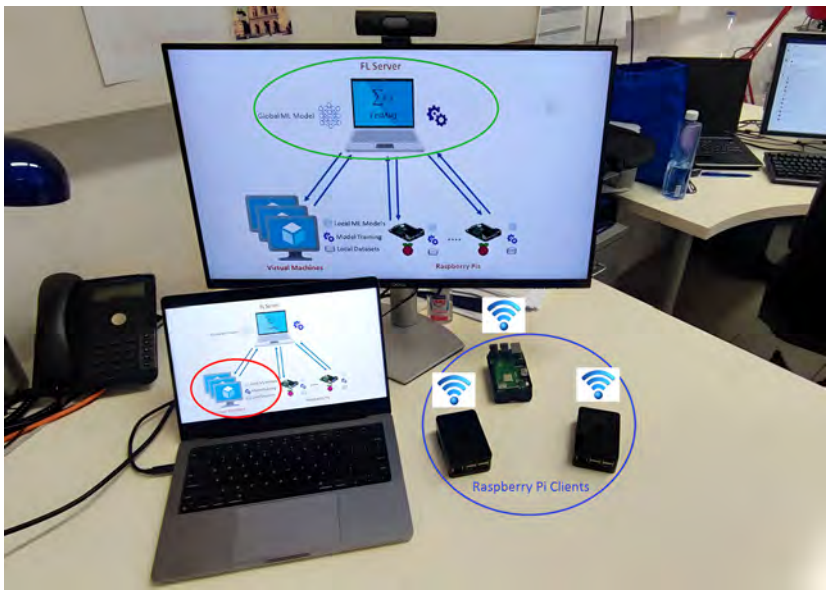

Figure 3. Experimental Setup Used During the FL Implementation.

4. Simulations and Performance Evaluations

In this section, with the help of the Python simulation environment, we analyze the performance of the FL framework proposed before. For a considered image classification problem, the CIFAR10 dataset is used. It contains 60,000 colorful pictures, with a resolution of 32 $\times$ 32 pixels, separated into 10 classifications with 6000 pictures in each category. The CIFAR10 incorporates 50,000 samples for training and the remaining 10,000 samples for verification. The images are randomly arranged while maintaining a perfectly uniform distribution of the classes. In the training set, each client has precisely 5000 images. In the experimental setup considered, two Raspberry Pi devices are considered as well as up to 8 virtual clients. In our setup, each virtual client was allocated an equal share of CPU resources on the 8-core host machine. This means that the CPU resources were evenly distributed among the virtual clients, ensuring a fair and consistent experimental environment. By allocating an equal portion of the available 8-core CPU to each virtual client, we maintained a balanced and representative simulation, allowing us to assess the FL framework's performance accurately. This approach ensures that the results obtained were not influenced by uneven resource distribution among the virtual clients, providing a reliable basis for our experimental findings. At first, the CIFAR10 training set was distributed among the clients. The 10 FL iterations are considered with 3 local training epochs. The server uses the test data of 10,000 CIFAR10 samples throughout the simulation. The accuracy of the model was determined using a metric that calculated the percentage of correct predictions among all predictions. The accuracy values presented in our figures are indeed normalized, ranging from 0 to 1, and not represented in percentages. For example, an accuracy value of 0.7 corresponds to 70 percent. Note that, to avoid the extensive training process we have limited the number of training iterations and the overall data size. However, this also upper-bounds the overall performance of DNN. In the experimental studies considered, DNN can only achieve an accuracy of up to 65%. However, the performance can be improved with additional resources, i.e., data samples, devices, training iterations, etc.

Figure 4, presents the performance of the FL framework considered in the basic settings, that is, a set of users having the same amount of data and onboard capabilities. This simulation is limited to Raspberry PI devices only. Both centralized and FL models are trained for 30 epochs. To have an adequate comparison over different training epochs, both models' performance in terms of accuracy is plotted with respect to the incremental values of the training epoch. It should be noted that in a considered simulation, the overall epochs represent the overall training process iterations, and therefore for the case of FL, the epochs are based on the number of FL iterations (rounds) times the local epoch performed. From Figure 4, it can be visualized that the FL framework can emulate the performance of a centralized approach in close proximity. Though FL can have a slightly reduced performance compared to the centralized case, the added advantage in terms of reduced communication overheads, and enhanced data privacy can be crucial advantages in the novel wireless scenarios. It also highlights that all the components of the proposed FL framework are configured properly and thus the platform is ready to perform the additional experimental steps.

In the next parts, to evaluate the performance of the models trained with FL, they are compared with a centralized learning benchmark with the same neural network. Some of the key variables considered during the experimentation are the number of clients participating in the simulation, the amount of data used for learning, and the selection and partitioning of the training data. Simulations are conducted to assess the impact of a single variable as well as different combinations. Additionally, performance differences between virtual clients implemented on Windows computers and hardware clients of Raspberry PI are carefully analyzed. The introduction of pre-trained models and alternative learning paradigms such as Transfer Learning (TL) are also considered.

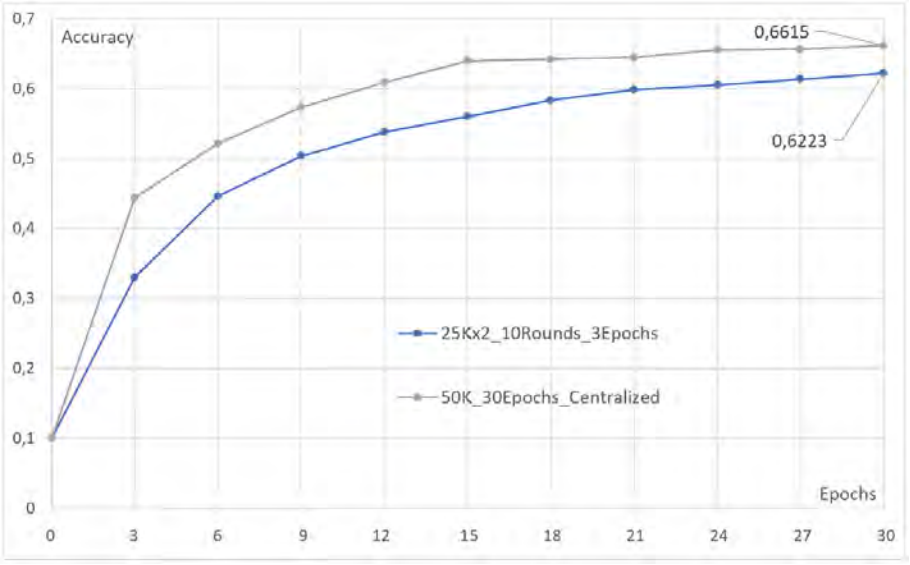

Figure 4. Accuracy of FL for 2 clients compared to the centralized benchmark vs. epochs.

4.1. Effect of a Cooling Mechanism

The excessive amount of heat generated by the computation hardware can have a severe impact on the environment and is underlined quite often. Such issues can also affect the performance of the device, which can impact the model training performance. For the case of FL, it is important to analyze the impact of such heating issues on the training performances given the involvement of a large number of sensory devices. Therefore, in Figure 5, we have presented the performance of two FL models with similar tasks and training environments. One of the FL studies involves the utilization of a cooling fan to reduce the heating issues of Raspberry Pi clients. As is evident from this figure, accuracy can be improved with the use of cooling devices, which can help to achieve model convergence rapidly. This highlights the importance of incorporating novel cooling mechanisms into end devices to enable efficient intelligent solutions in wireless networks. As such, all subsequent simulations were conducted in the presence of a cooling fan.

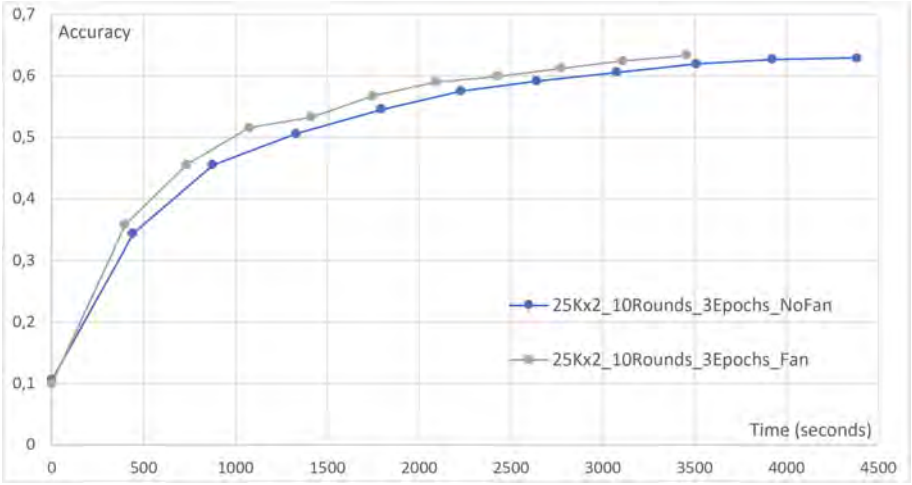

Figure 5. The effect of a cooling fan on the accuracy of training.

4.2. Heterogeneous Client Compensation

One of the significant challenges in implementing distributed learning techniques such as FL is the presence of heterogeneous client devices with different amounts of resources, i.e., computational capabilities, training data, etc. In a considered FL implementation two sets of clients are considered. While implementing the FL solutions, a significant disparity

in performance was promptly observed between clients operating as virtual machines and those operating on Raspberry Pis. In the case of the traditional FL approach with heterogeneous sets of clients, the server node waits until it receives the updates from all the clients in question. In a considered FL setup, Specifically, it has been observed that the server waits until even the Raspberry Pi clients complete their training, even after the most powerful clients have finished their training. We have implemented two approaches to mitigate this waiting time and make the best use of the highest-performing clients. The first approach involves training the best devices for several rounds before transmitting the model to the server. In contrast, the second approach involves training all clients with the same number of periods but using more data for high-performance clients. Raspberry Pi devices with their limited capabilities often take longer duration to communicate their model updates adding a large amount of communication overhead. On the other hand, virtual clients with a significant amount of resources are able to conclude the learning process promptly, they suffer due to the poor behavior of hardware clients. To counter this issue, we have adopted two different strategies. In the first case, we have normalized the FL iteration time by inducing the harsh local training conditions over the virtual machine-based clients by increasing the number of local epochs performed. With this approach, instead of staying in an idle state and waiting for the parameter updates from the slow-performing clients, the virtual clients try to optimize the local model performance through more training. In the second approach, we have normalized the FL iteration time through different data splits. In this case, each node performs the same amount of training epoch; however, the number of data samples considered at a virtual machine is significantly higher than the Raspberry Pi clients.

The FL data split can be performed with different methods. In Figure 6, we have presented the FL model performance with asymmetric data split between heterogeneous clients. Each subfigure includes a centralized benchmark with 50 K samples, an FL approach with a 4:1 split, and another FL case with a 3:2 split. The subfigure on the left is based on the deterministic data split approach, where the repetition of data samples at different nodes is avoided. While in the subfigure on the right, a random data selection approach is adopted, without taking into account the repetitions of data samples at different devices. There is a significant gap between the performance of deterministic and random selection approaches. The deterministic approach can improve the accuracy of the FL model by up to 3.3% compared to the random case.

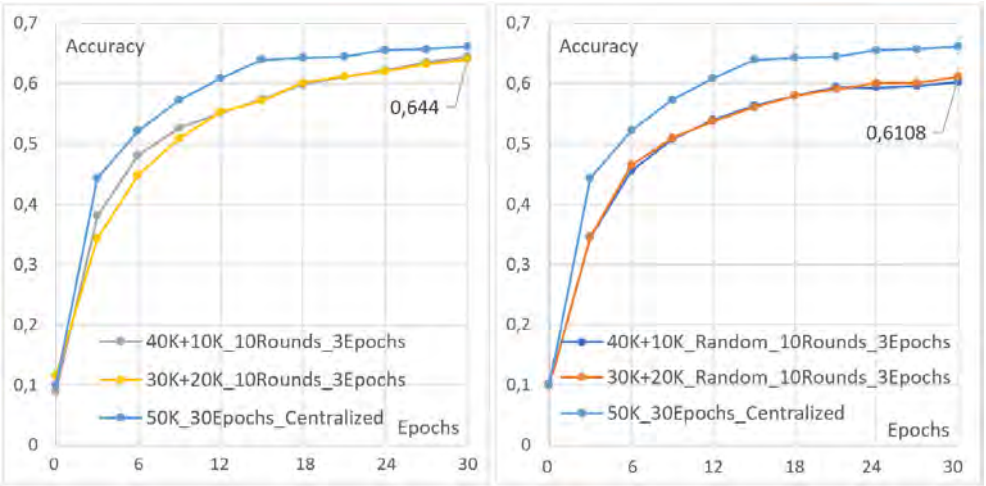

Figure 6. Simulations with asymmetric data distribution, without and with random selection compared to the Centralized Benchmark.

In the next case, we varied the local training epoch over different clients to analyze the performance. In this case, all devices use the same amount of training samples, while varying the nature of the local training process. In particular virtual machine-based clients

perform more local epochs compared to the hardware nodes, before communicating their updates to the parameter server. Figure 7 compares the accuracy of different data split options along with the differing local training processes. In this case, asynchronous data split achieves higher accuracy, compared to the two FL cases where the same amount of data is used by the FL clients while normalizing the FL process time through the adaptive local training operations. To verify this trend and lend it further credence, it was deemed necessary to perform several similar tests. These tests were carefully arranged so that their execution times were identical, thus rendering it feasible to obtain a more precise comparison in the time domain. This is demonstrated in Figure 8, which compares the two aforementioned methods for distributing the workload between two heterogeneous clients through several simulations. These simulations entail the following conditions:

- Two clients, each with 25,000 images. The first client undergoes two local epochs, while the second undergoes eight.
- Two clients, each with five local epochs. The first client is assigned 10,000 images, while the second is assigned 40,000.
- Two clients, each with 25,000 images. The first client undergoes three local epochs, while the second undergoes twelve.
- Two clients, each with seven local epochs. The first client is assigned 10,000 images, while the second is assigned 40,000.

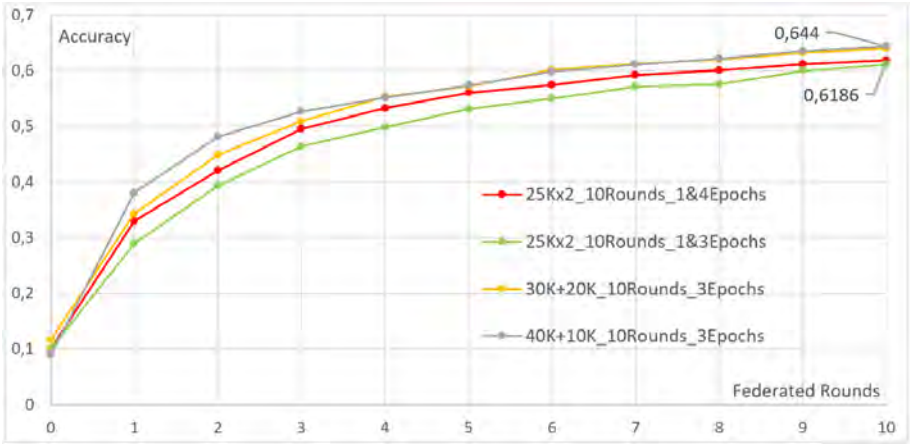

Figure 7. Accuracy with asymmetric distribution of data vs different numbers of local epochs.

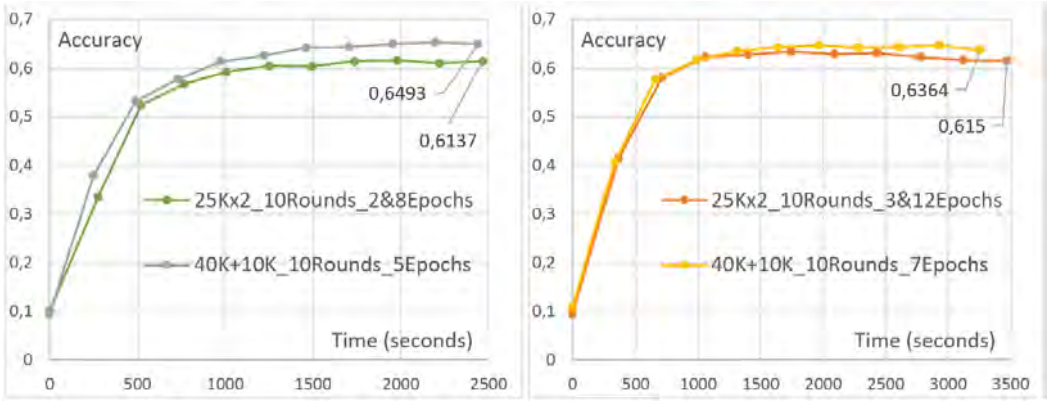

Figure 8. Accuracy for asymmetric data distribution vs. different numbers of local epochs, in time domain.

In our study, we intentionally introduced inhomogeneous and asymmetric data distribution to mimic real-world scenarios where data across users can vary significantly. Specifically in Figure 8, the yellow line corresponds to a balanced data distribution, where the dataset was evenly split between users (i.e., Raspberry Pis), with each user having 20,000 images. In contrast, the orange line illustrates an uneven data distribution. In this scenario, one user was allocated 10,000 images, while the other user had 40,000 images. This intentional variation allows us to evaluate the FL framework's ability to handle disparate data quantities among users. In all cases, the aforementioned trend is consistently observed. This underscores the efficiency of adapting asynchronous data splitting across heterogeneous FL clients, rather than modifying the overall training process to enforce synchronization of FL updates from diverse devices.

Another issue that frequently impacts the DNN performance is the concept of overfitting especially for the case of unbalanced datasets with a large number of local training epochs. While modifying the local training process for the case of FL with heterogeneous nodes, the data available at different nodes should be taken into account. When examining the final parts of the two simulations, in Figure 9, it is possible to observe the effect of overfitting. The accuracy has somewhat lessened due to excessive training of the model on a small amount of data, leading to a decrease in universality for examples that were not utilized for training.

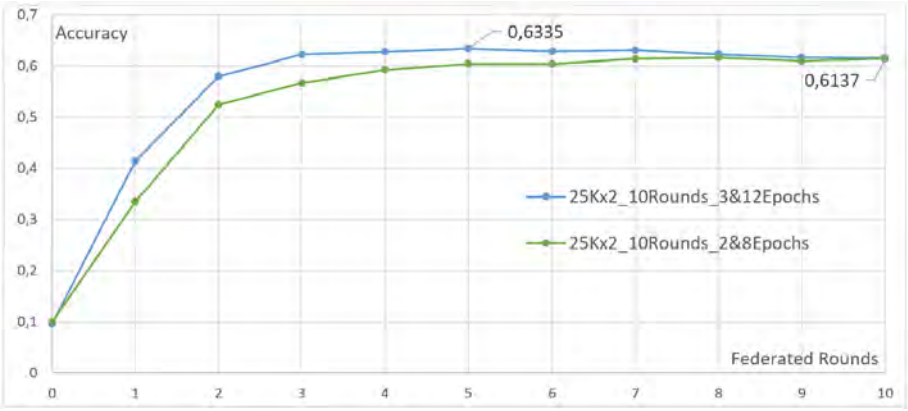

Figure 9. Overfitting for different numbers of local epochs.

It is worth mentioning that compensating for performance by distributing unequal amounts of data between devices is a challenging approach to implement. This is because it may not be possible to control the volumes of data collected by different clients. Conversely, increasing the workload of the highest-performing devices by increasing their local epochs can be easily achieved with communications from the server.

After defining the scenarios mentioned above, further exploration was conducted to determine the impact of a random selection of data compared to an ordered split without overlaps. It is important to note that the CIFAR10 dataset samples are inherently unordered, with images following one another irrespective of their class.

The initial approach involved dividing the dataset among multiple clients without repetition, resulting in a total of 50,000 samples between all clients, which is the complete training data. Imagine a circumstance where a pair of patrons are tasked with handling the preparation assortment of CIFAR10, with one individual managing the initial 25,000 samples and the other in command of the residual 25,000 samples. These two data groups do not overlap, thus utilizing the entire training set.

Expanding this approach to five clients would require dividing the data set into five categories, each with 10,000 samples, with no overlaps or sample duplications. The second approach, however, was based on the random selection of samples from the dataset, allowing for a single data piece to be chosen several times within the same client. Moreover, data could be common to different clients, even if the sum of the samples from both clients

exceeds the total number of samples from the training set. Thus, this method may yield a generally substandard model performance that aligns more with actual situations.

It is essential to consider that performance may vary from simulation to simulation, even with the same conditions, due to the random variables of these tests. As a result, the tests involving random samples were repeated multiple times, and the results presented represent their average. The impact of random selection, with identical data amounts and epoch numbers, is evident in Figure 10, which compares two centralized learnings.

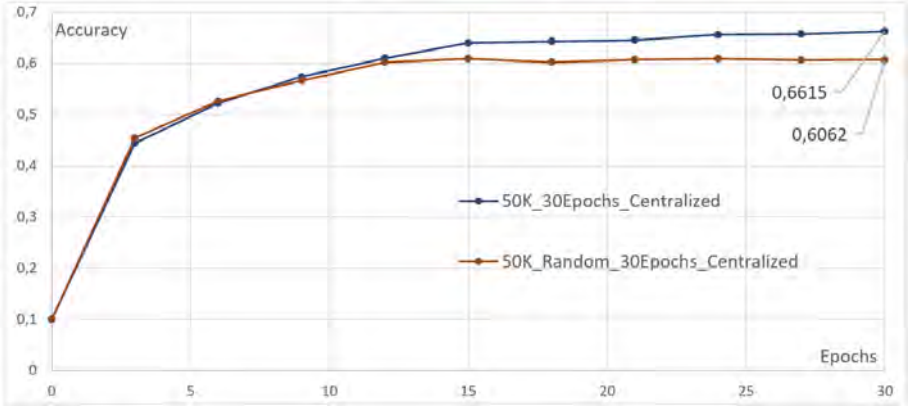

Figure 10. Centralized Learning with and without random data selection.

The aforementioned phenomenon is similarly observed in a federated scenario, as illustrated in Figure 11 by comparing two basic FLs involving two clients, each containing 25,000 images. The likelihood of encompassing a greater portion of the dataset and subsequently enhancing the ultimate accuracy increases as the number of randomly selected training samples increases.

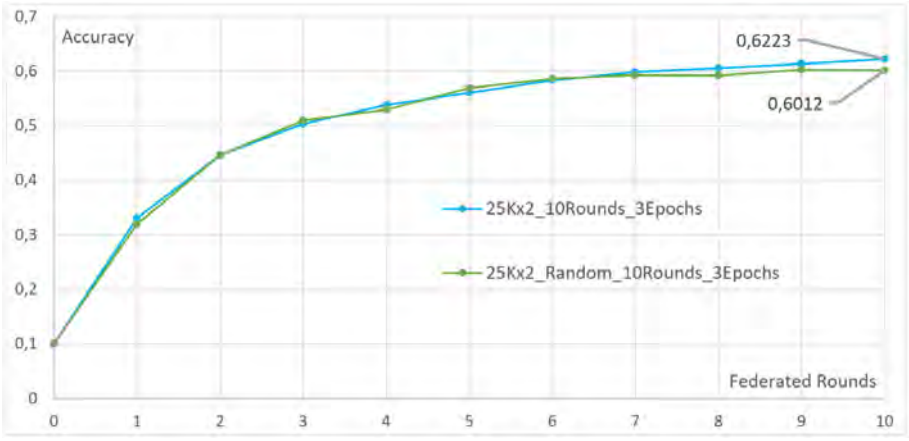

Figure 11. Federated Learning of 2 clients with and without random data selection.

As the quantity of randomly selected samples utilized for training increases, the probability of encompassing a more significant portion of the dataset also increases. This subsequently results in an enhancement of the final accuracy. This correlation is visually depicted in Figure 12, showcasing three FL evaluations conducted on two devices. The number of samples on each client has been progressively augmented from one simulation to the next, positively impacting the accuracy metric at the conclusion of each federated round.

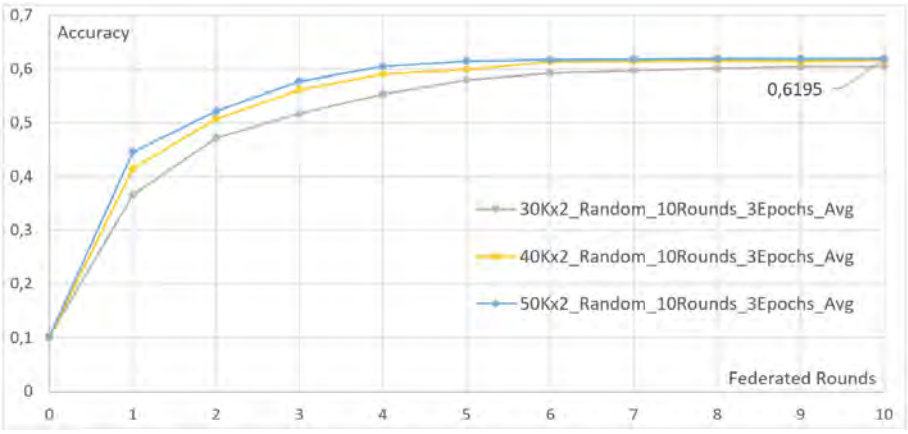

Figure 12. FL with different amounts of randomly chosen samples for two clients.

4.3. Increasing the Number of Clients

The subsequent phase, aimed at enhancing the observance of FL and its scalability, entailed increasing the number of clients participating in the simulation to a maximum of 10. Of these, 2 were Raspberry clients, and 8 were virtual clients, coexisting with the server on the computer. The 10-client threshold could not be exceeded due to the restricted RAM on the laptop and the considerable resources required to train neural networks. Nevertheless, this number of clients proved adequate in inducing a noteworthy decline in the model's performance as the number of devices involved escalated. In addition, it can be seen that the accuracy calculated by the server of the global model after each federated round is progressively sluggish as the number of clients used grows.

Figure 13 portrays the accuracy value for each federated round for five different simulations, each divided equally across all clients, using the internal training set (50,000 images). The specific parameters for each test are as follows: two clients with 25,000 samples each, four clients with 12,500 samples each, five clients with 10,000 samples each, eight clients with 6250 samples each, and ten clients with 5000 samples each. The simulation employing only two clients is the best, whereas the test with ten devices yields the worst performance, requiring 40 federated rounds before achieving accuracy comparable to the former. However, it should be noted that, in the first case, Raspberries must operate with 25,000 samples each, resulting in relatively slower local periods, while in the second case, each client uses only 5000 samples, thereby completing its local epochs much faster.

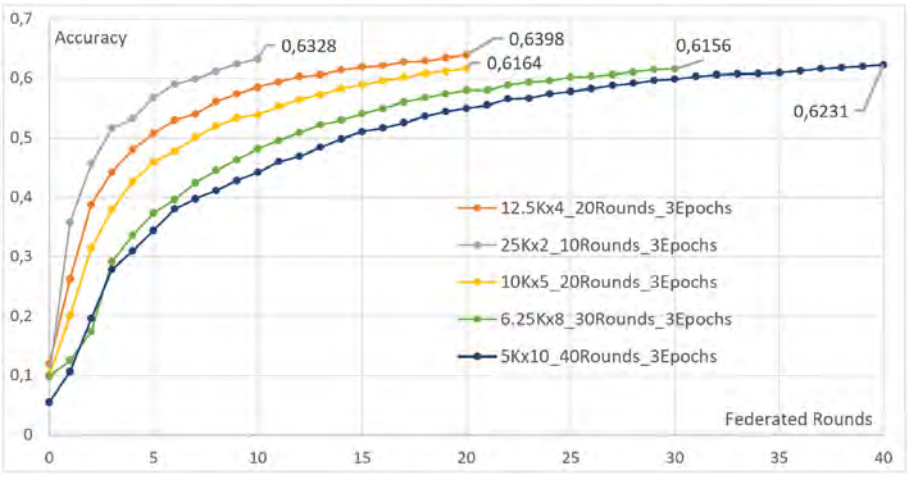

Figure 13. FL with different amounts of randomly distributed samples among different number of clients.

Examining Figure 14, representing the same performance while in the time domain, it is evident that the actual difference in performance between the simulations is much lower and sometimes even nonexistent. For instance, the test with ten clients completed its 40 rounds at a time, close to the tenth round of the test with only two clients. The same applies to the other curves, with the simulation involving four clients overlapping perfectly with that of two. This implies that, in the context of a system with limited computational capacity, as represented by the Raspberry Pi, a federated approach generates relatively similar performance to centralized ones.

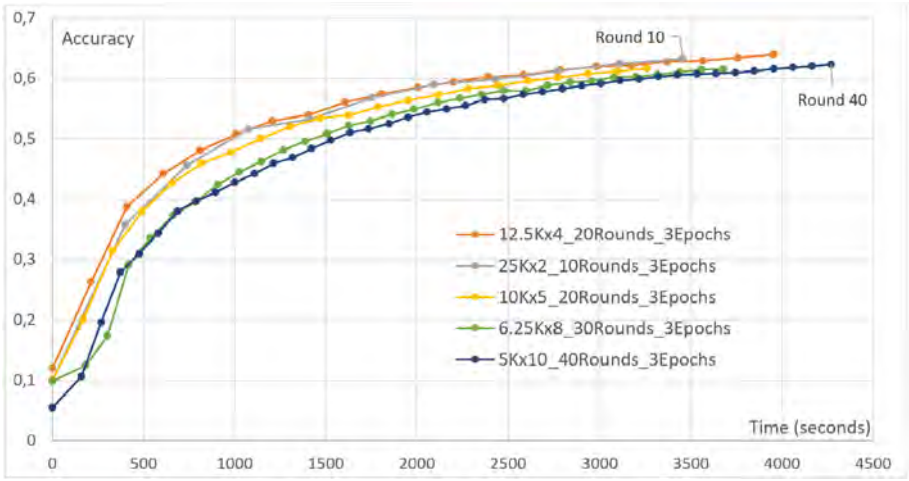

Figure 14. Accuracy vs. the number of rounds for different numbers of clients.

A similar pattern can be witnessed even with random samples, as illustrated in the two simulations depicted in Figure 15. The clients work with 30,000 and 12,000 images randomly selected from the dataset, and the algorithm trained with five clients obtains lower accuracy per round. However, since these simulations are completed twice as fast as the two-client simulation, the result is that the accuracy values are very similar moment by moment.

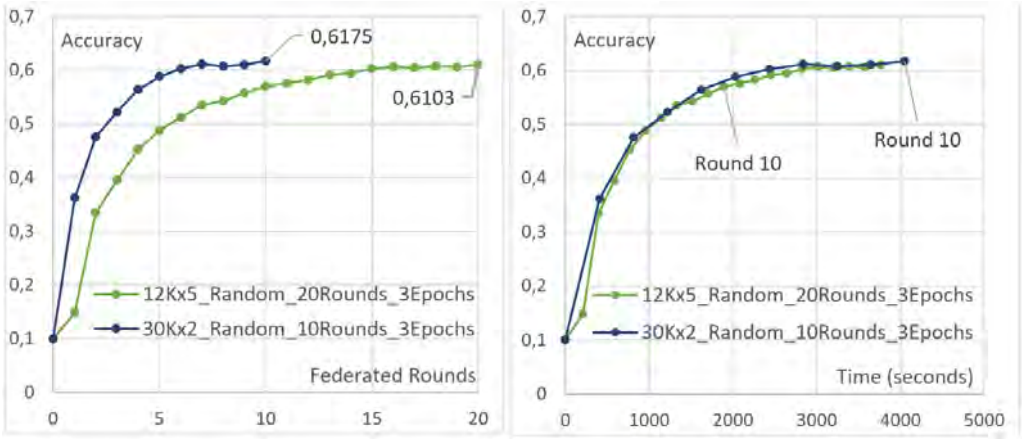

Figure 15. Accuracy with random samples and different numbers of clients.

4.4. Effect of Pretraining

Upon completion of the previous phase, the training framework was further explored, emphasizing the introduction of pre-trained models. In all tests conducted thus far, the initial accuracy starts at approximately 0.1 (i.e., 10%), which aligns with a randomly initialized model. In this FL implementation, at the beginning of each simulation, the server randomly selects a client to transmit its original model, a neural network with randomly assigned node weights, generating a very low accuracy of approximately 10%. This model is then

disseminated from the server to all clients to establish a common starting point for training. Alternatively, it may be feasible to directly preserve a fundamental model on the server that is conveyed to all users during the initial stage. This method allows for using a pre-trained neural network already on the server, resulting in enhanced performance for FL.

For this specific scenario, the model pre-training was conducted centrally on 30,000 samples randomly selected by CIFAR10, with data processing for 3, 6, or 9 periods, leading to three pre-trained models with varying levels of accuracy. This approach observed how pretraining at different intensities contributes to FL's performance. As demonstrated in Figure 16, each test has a distinct initial accuracy value, followed by training through a straightforward implementation of federated learning with two clients, each with half a dataset and three local epochs per round. The impact of pretraining is discernible in the initial rounds, where trained simulations exhibit a significant advantage over cases without pretraining. This discrepancy weakens as the federated rounds continue until all curves converge around the tenth round, resulting in a final accuracy value that is relatively high, especially considering that the previous centralized benchmark on the internal training set had a score of 65%.

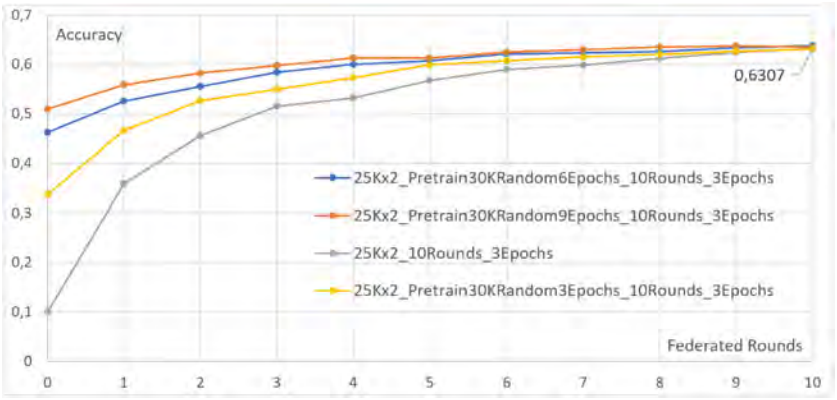

Figure 16. Accuracy with 2 clients and different pretraining levels.

Similar examinations were replicated in a federated context comprising five clients, each trained on a fifth of the dataset to extend the findings. Once again, as can be seen in Figure 17, a trend similar to the previous one is discernible, with a substantial effect of pretraining in the initial stages that diminishes until the curves converge. The incorporation of pre-trained models thus represents a commendable strategy to accelerate the federated learning of the network. This technique is particularly advantageous in situations with limited time available for training, as in this case, FL could be stopped before converging to maximum accuracy, and pretraining would bring significant benefits, as observed in the first part of the graph.

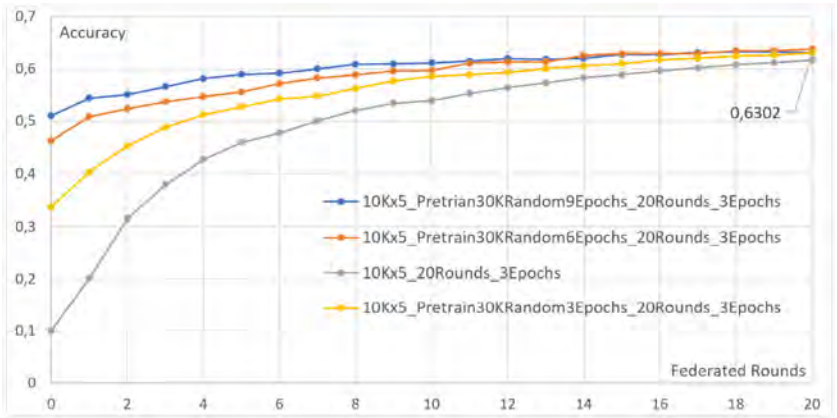

Figure 17. Simulations with 5 clients and different pretraining levels.

To further emphasize the impact of pretraining, FL was conducted again with 10 clients who had previously exhibited reduced performance compared to other simulations. However, this time a starting model already trained with 30,000 random samples for six periods was used. The outcome is shown in Figure 18, where the new simulation has substantially higher accuracy than the previous test with 10 clients and even surpasses the scenario with only two devices. This exemplifies how one can compensate for the decline in the accuracy of FL due to the increase in the number of devices involved by incorporating pretraining.

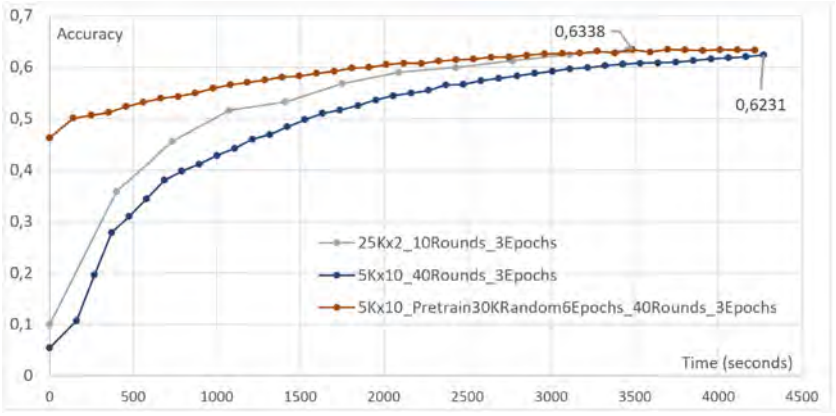

Figure 18. 2 simulations with 10 clients with or without pretraining vs. simulation with 2 clients without pretraining.

However, we must recognize the deficiencies of this approach. Pretraining a model demands time and energy. Ideally, the optimal solution would be to conduct the pretraining directly on the server in a centralized manner. In this case, the reduced number of periods required to obtain a good starting model and the high computing capacity of the server would render the pretraining time almost negligible compared to the duration of the subsequent federated training. Nonetheless, this solution may not be feasible in a practical scenario as it implies the presence of a certain amount of training data directly on the server when, in several cases, FL is chosen precisely to circumvent the transfer of the local data collected by the various devices. A possible alternative could be the utilization of a more generic dataset for pretraining, then leaving clients with the task of integrating the model's specific features with their local data, akin to what occurs for TL.

4.5. Transfer Learning

Drawing inspiration from the aforementioned analogy of TL, executing its functionality in the present framework has been feasible. However, it is imperative to specify that the measures implemented to achieve this outcome do not facilitate the attainment of the same level of automation as federated simulations. The practical principle is akin to pretraining, wherein an initial model is trained through centralized machine learning. Subsequently, this model is disseminated to various devices that conduct local training with their unique dataset.

In the first part of Figure 19, the fundamental model training is depicted, followed by the curves of multiple models obtained by the clients. The two models produced in the first scenario are almost identical, as both clients have 25,000 images of data and have been trained for the same number of epochs.

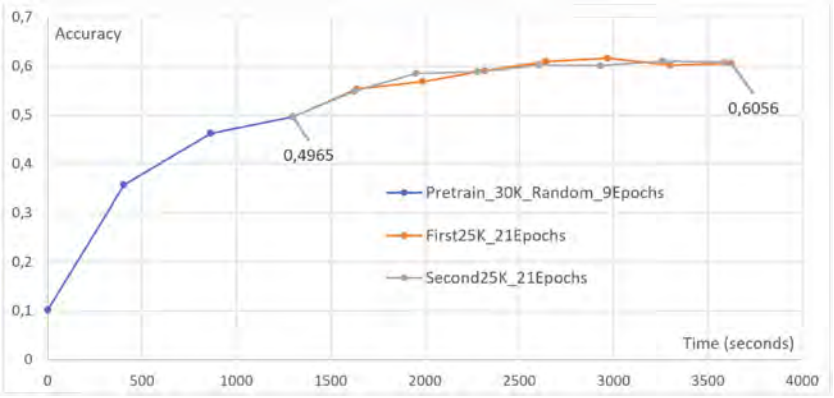

Figure 19. TL with 2 clients.

In contrast, in Figure 20, differences in terms of accuracy and training time between the three devices are apparent, which is expected since all the clients have trained for 21 periods. However, each client had varying sample sizes available, with the first client having 10,000, the second having 20,000, and the third having 30,000 elements of CIFAR10. Thus, clients with more data will exhibit slower periods that guarantee greater accuracy.

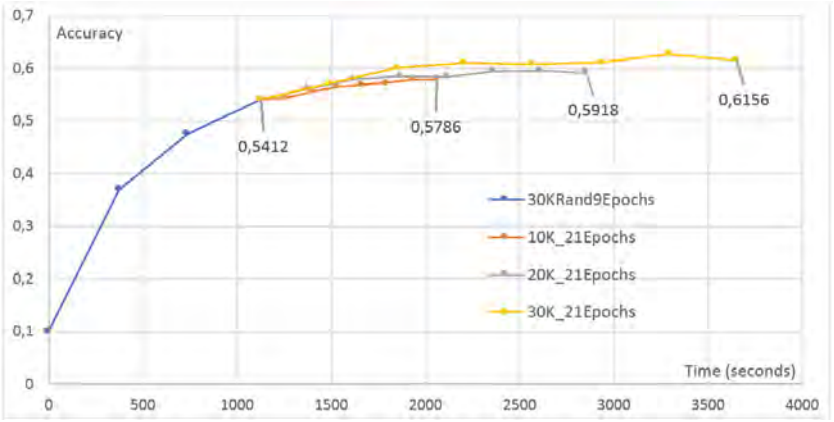

Figure 20. TL with 3 clients.

5. Discussion

In this study, we conducted an in-depth exploration of Federated Learning (FL) methodology across heterogeneous nodes, offering critical insights into its application within the context of 6G technology. Our investigation revolved around a hardware-software integrated FL model, ingeniously leveraging a combination of Raspberry Pi devices and virtual machines as FL clients, each equipped with unique datasets sourced from the CIFAR10 dataset, a widely accepted benchmark in image classification. The experiments were meticulously designed to mirror real-world scenarios, addressing multifaceted challenges including varying computation resources, uneven dataset distributions, and the heating issues inherent in wireless devices.

A pivotal aspect of our research involved examining the impact of cooling mechanisms on training accuracy, elucidated in detail in Section 4.1. The insights garnered from Figure 5 underscore the significance of cooling devices in expediting model convergence, highlighting their practical relevance, especially in resource-constrained environments. Additionally, we delved into the complexities of heterogeneous client compensation, meticulously examining asymmetric data distribution scenarios, both with and without random selection, compared to a Centralized Benchmark, as delineated in Section 4.2 (Figures 6–8). These analyses illuminated the nuanced dynamics of FL performance, emphasizing the intricate balance required in distributing training data among disparate nodes. Furthermore, our study probed the issue of overfitting in FL, a critical concern often encountered in decentral-

ized learning paradigms. Through Figure 9, we identified the challenge and subsequently addressed it by incorporating random selection strategies, showcased in Figures 10–12, thereby mitigating overfitting risks while optimizing model generalization. A comprehensive exploration into the scalability of FL was presented in Section 4.3, analyzing the impact of increasing the number of users on the system's performance (Figures 13–15). This analysis provided valuable insights into FL's scalability potential, crucial for its adoption in large-scale, dynamic environments. The effectiveness of pretraining in enhancing accuracy rates was explored in Section 4.4, revealing significant improvements showcased in Figures 16–18. Pretraining emerged as a powerful technique, elevating the model's performance and showcasing its potential for optimizing FL outcomes. Finally, in Section 4.5, we delved into Transfer Learning's potential, evaluating its impact with varying numbers of clients (Figures 19 and 20). The results underscored Transfer Learning's capacity to enhance FL model performance, particularly when faced with diverse client configurations.

6. Conclusions

In this work, we have investigated the performance of the FL method including a group of heterogeneous nodes. In particular, a hardware-software integrated FL model is developed by using a set of Raspberry PI devices and virtual machines acting as an FL client with their datasets. Performance is analyzed for the case of an image classification problem with a widely known CIFAR10 dataset. Given the importance of distributed intelligence with heterogeneous wireless nodes in the upcoming 6G technology, a set of experiments are performed to analyze the FL performance in different cases. Main issues such as differing computation resources, uneven distributions of datasets, and heating issues of wireless devices were considered while performing the experiments. In addition, novel technologies such as the users of pre-trained networks for knowledge transfer, were also considered. A more pro-analysis and concluding remarks are also presented during the discussion of simulation results. This study can be highly useful when considering the deployments of FL methods over heterogeneous 6G environments to enable large-scale, connected, cost-efficient, and reliable distributed intelligence.

In conclusion, our study's multifaceted approach, spanning cooling mechanisms, heterogeneous compensation strategies, overfitting mitigation, scalability analyses, and the integration of pretraining and Transfer Learning, provides a holistic understanding of FL's dynamics across heterogeneous nodes. These nuanced findings not only contribute significantly to the academic discourse but also hold practical implications for real-world 6G deployments. By illuminating the complexities and offering viable solutions, our research empowers the seamless integration of FL in diverse, large-scale, connected, cost-efficient, and reliable distributed intelligence systems, laying the foundation for the future of intelligent wireless networks.

Author Contributions: Conceptualization, D.T. and S.S.S.; methodology, S.S.S.; software, L.R. and D.N.; validation, L.R., S.S.S. and D.N.; formal analysis, S.S.S. and D.N.; investigation, L.R.; resources, D.T.; data curation, L.R.; writing—original draft preparation, D.N.; writing—review and editing, D.T., D.N, and S.S.S.; visualization, L.R.; supervision, D.T.; project administration, D.T.; funding acquisition, D.T. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially supported by the ECOSISTER project funded under the National Recovery and Resilience Plan (NRRP), Mission 04 Component 2 Investment 1.5—NextGenerationEU, Call for tender n. 3277 dated 30 December 2021, Award Number: 0001052 dated 23 June 2022 and by the European Union under the Italian National Recovery and Resilience Plan (NRRP) of NextGenerationEU, partnership on "Telecommunications of the Future" (PE00000001—program "RESTART").

Data Availability Statement: Data available on request due to restrictions e.g., privacy or ethical.

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

References

1. Letaief, K.B.; Shi, Y.; Lu, J.; Lu, J. Edge Artificial Intelligence for 6G: Vision, Enabling Technologies, and Applications. *IEEE J. Sel. Areas Commun.* **2022**, *40*, 5–36. [CrossRef]
2. Nguyen, D.C.; Ding, M.; Pathirana, P.N.; Seneviratne, A.; Li, J.; Niyato, D.; Dobre, O.; Poor, H.V. 6G Internet of Things: A Comprehensive Survey. *IEEE Internet Things J.* **2022**, *9*, 359–383. [CrossRef]
3. *6G Technology Overview*, 2nd ed.; one6G White Paper; 2022. Available online: https://one6g.org/download/2699/ (accessed on 27 September 2023).
4. Tang, F.; Mao, B.; Kawamoto, Y.; Kato, N. Survey on Machine Learning for Intelligent End-to-End Communication Toward 6G: From Network Access, Routing to Traffic Control and Streaming Adaption. *IEEE Commun. Surv. Tutor.* **2021**, *23*, 1578–1598. [CrossRef]
5. Muscinelli, E.; Shinde, S.S.; Tarchi, D. Overview of Distributed Machine Learning Techniques for 6G Networks. *Algorithms* **2022**, *15*, 210. [CrossRef]
6. Shinde, S.S.; Bozorgchenani, A.; Tarchi, D.; Ni, Q. On the Design of Federated Learning in Latency and Energy Constrained Computation Offloading Operations in Vehicular Edge Computing Systems. *IEEE Trans. Veh. Technol.* **2022**, *71*, 2041–2057. [CrossRef]
7. Shinde, S.S.; Tarchi, D. Joint Air-Ground Distributed Federated Learning for Intelligent Transportation Systems. *IEEE Trans. Intell. Transp. Syst.* **2023**, *24*, 9996–10011. [CrossRef]
8. Duan, Q.; Huang, J.; Hu, S.; Deng, R.; Lu, Z.; Yu, S. Combining Federated Learning and Edge Computing Toward Ubiquitous Intelligence in 6G Network: Challenges, Recent Advances, and Future Directions. *IEEE Commun. Surv. Tutor.* **2023**, *in press*. [CrossRef]
9. Morocho-Cayamcela, M.E.; Lee, H.; Lim, W. Machine Learning for 5G/B5G Mobile and Wireless Communications: Potential, Limitations, and Future Directions. *IEEE Access* **2019**, *7*, 137184–137206. [CrossRef]
10. Praveen Kumar, D.; Amgoth, T.; Annavarapu, C.S.R. Machine learning algorithms for wireless sensor networks: A survey. *Inf. Fusion* **2019**, *49*, 1–25. [CrossRef]
11. Fontanesi, G.; Ortíz, F.; Lagunas, E.; Baeza, V.M.; Vázquez, M.; Vásquez-Peralvo, J.; Minardi, M.; Vu, H.; Honnaiah, P.; Lacoste, C.; et al. Artificial Intelligence for Satellite Communication and Non-Terrestrial Networks: A Survey. *arXiv* **2023**, arXiv:2304.13008.
12. Lee, H.; Lee, S.H.; Quek, T.Q.S. Deep Learning for Distributed Optimization: Applications to Wireless Resource Management. *IEEE J. Sel. Areas Commun.* **2019**, *37*, 2251–2266. [CrossRef]
13. Huang, J.; Wan, J.; Lv, B.; Ye, Q.; Chen, Y. Joint Computation Offloading and Resource Allocation for Edge-Cloud Collaboration in Internet of Vehicles via Deep Reinforcement Learning. *IEEE Syst. J.* **2023**, *17*, 2500–2511. [CrossRef]
14. Song, H.; Liu, L.; Ashdown, J.; Yi, Y. A Deep Reinforcement Learning Framework for Spectrum Management in Dynamic Spectrum Access. *IEEE Internet Things J.* **2021**, *8*, 11208–11218. [CrossRef]
15. Nayak, P.; Swetha, G.; Gupta, S.; Madhavi, K. Routing in wireless sensor networks using machine learning techniques: Challenges and opportunities. *Measurement* **2021**, *178*, 108974. [CrossRef]
16. Yang, Z.; Chen, M.; Saad, W.; Hong, C.S.; Shikh-Bahaei, M. Energy Efficient Federated Learning Over Wireless Communication Networks. *IEEE Trans. Wirel. Commun.* **2021**, *20*, 1935–1949. [CrossRef]
17. Jiang, J.C.; Kantarci, B.; Oktug, S.; Soyata, T. Federated Learning in Smart City Sensing: Challenges and Opportunities. *Sensors* **2020**, *20*, 6230. [CrossRef] [PubMed]
18. Matthiesen, B.; Razmi, N.; Leyva-Mayorga, I.; Dekorsy, A.; Popovski, P. Federated Learning in Satellite Constellations. *IEEE Netw.* **2023**, 1–16. *in press*. [CrossRef]
19. Younus, M.U.; Khan, M.K.; Bhatti, A.R. Improving the Software-Defined Wireless Sensor Networks Routing Performance Using Reinforcement Learning. *IEEE Internet Things J.* **2022**, *9*, 3495–3508. [CrossRef]
20. Dewangan, D.K.; Sahu, S.P. Deep Learning-Based Speed Bump Detection Model for Intelligent Vehicle System Using Raspberry Pi. *IEEE Sens. J.* **2021**, *21*, 3570–3578. [CrossRef]
21. Cicceri, G.; Tricomi, G.; Benomar, Z.; Longo, F.; Puliafito, A.; Merlino, G. DILoCC: An approach for Distributed Incremental Learning across the Computing Continuum. In Proceedings of the 2021 IEEE International Conference on Smart Computing (SMARTCOMP), Irvine, CA, USA, 23–27 August 2021; pp. 113–120. [CrossRef]
22. Mills, J.; Hu, J.; Min, G. Communication-Efficient Federated Learning for Wireless Edge Intelligence in IoT. *IEEE Internet Things J.* **2020**, *7*, 5986–5994. [CrossRef]
23. Farkas, A.; Kertész, G.; Lovas, R. Parallel and Distributed Training of Deep Neural Networks: A brief overview. In Proceedings of the 2020 IEEE 24th International Conference on Intelligent Engineering Systems (INES), Reykjavík, Iceland, 8–10 July 2020; pp. 165–170. [CrossRef]
24. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. [CrossRef]

25. Hong, C.S.; Khan, L.U.; Chen, M.; Chen, D.; Saad, W.; Han, Z. *Federated Learning for Wireless Networks*; Springer: Singapore, 2022.
26. Zhang, Z.; Zhang, Y.; Guo, D.; Zhao, S.; Zhu, X. Communication-efficient federated continual learning for distributed learning system with Non-IID data. *Sci. China Inf. Sci.* **2023**, *66*, 122102. [CrossRef]