Article

Not peer-reviewed version

# Federated Learning in Dynamic and Heterogeneous Environments: Advantages, Performances, and Privacy Problems

Fabio Liberti , Davide Berardi * , Barbara Martini

Posted Date: 29 August 2024

doi: 10.20944/preprints202408.2125.v1

Keywords: Privacy; Federated Machine Learning; Edge Computing; Ubiquitous Intelligence

 doi:10.20944/preprints202408.2125.v1



*Article*

# Federated Learning in Dynamic and Heterogeneous Environments: Advantages, Performances, and Privacy Problems

**Fabio Liberti [1,†], Davide Berardi [2,†,*] and Barbara Martini [3,†]**

[1] fabio.liberti@studenti.unimercatorum.it
[2] davide.berardi@unimercatorum.it
[3] barbara.martini@unimercatorum.it
* Correspondence: davide.berardi@unimercatorum.it
† Universitas Mercatorum, Department of Science and Engineering, Rome, Italy.

**Abstract:** Federated Learning (FL) represents a promising distributed learning methodology, particularly suitable for dynamic and heterogeneous environments characterized by the presence of Internet of Things (IoT) devices and Edge Computing infrastructures. In this context, FL allows you to train machine learning models directly on edge devices, mitigating data privacy concerns and reducing latency due to transmitting data to central servers. However, the heterogeneity of computational resources, the variability of network connections, and the mobility of IoT devices pose significant challenges to the efficient implementation of FL. This work explores advanced techniques for dynamic model adaptation and heterogeneous data management in edge computing scenarios, proposing innovative solutions to improve the robustness and efficiency of federated learning. We present an innovative solution based on Kubernetes which enables the fast application of FL models to Heterogeneous Architectures. Experimental results demonstrate that our proposals can improve the performance of FL in IoT and edge environments, offering new perspectives for the practical implementation of decentralized intelligent systems.

**Keywords:** Privacy; Federated Machine Learning; Edge Computing; Ubiquitous Intelligence

## 1. Introduction

Federated Learning (FL) is a new approach to machine learning that allows you to train models on data distributed across different locations, without the need to centralize them. In this way, the information contained in large quantities of data can be exploited without compromising the privacy and security of the data itself. Federated Learning (FL) proves to be an ideal paradigm to collaboratively exploit the enormous amounts of data from devices located around the world, Internet of Things (IoT), while preserving privacy and security. Federated learning (FL) and edge computing are complementary technologies for distributed AI. FL allows edge devices to train models collaboratively, without sending all the data to the cloud. This reduces latency, improves privacy, and increases computational efficiency. In this work we focus on addressing the following Research Questions through the implementation of a distributed and federated machine learning software suite[1]:

**Q1** Which are the impacts of heterogeneous environments in Federated Learning?
**Q2** How can the federated learning scenario benefit from different systems and architectures?
**Q3** Which are the privacy implications of using federated learning?
**Q4** How can a cluster management system (i.e. Kubernetes) make the application of heterogeneous machine learning models feasible and more easy to do?

We argue that, replying to these research questions, we could create a clear view on the distributed machine learning scenario. This would highlight weak points and strengths of such methods.

[1] Track changes and periodical updates are available on the repository: https://github.com/FabioLiberti/DHFLPL

 doi:10.20944/preprints202408.2125.v1



### 1.1. Federated Learning and Edge Computing in the Internet of Things

**Ubiquitous intelligence.** Ubiquitous intelligence refers to intelligent systems that are pervasive and ubiquitous, integrated into the surrounding environment. These systems use sensors, mobile devices, advanced network technologies and artificial intelligence algorithms to provide advanced services and functionality without requiring direct user intervention. The goal is to create an intelligent environment that can anticipate user needs and respond proactively, improving the overall experience. This pervasive intelligence is fundamental in contexts such as smart cities, the Internet of Things (IoT) and new generation networks such as 6G, where the ability to provide services in real time and in a distributed manner is essential [1,2].

**Containers in Federated Learning.** Using containers, such as *Docker and Kubernetes*, in Federated Learning offers numerous benefits. Docker allows you to create isolated, reproducible environments for each participating node, ensuring consistency and ease of deployment. Kubernetes, on the other hand, facilitates the management of containers at scale, automating the deployment, scaling, and operation of containerized applications. This approach increases the scalability and resilience of the Federated Learning system, allowing it to easily manage many distributed nodes. Additionally, containers improve security by isolating processes and limiting access to resources. Integration with Continuous Integration/Continuous Deployment (CI/CD) tools further simplifies system upgrades and maintenance, ensuring that all participants are running the latest software versions [1,3]. These advantages make containers an ideal solution for implementing Federated Learning in heterogeneous, large-scale environments.

## 2. Related Work

### 2.1. Dynamic Federated Learning

Federated learning (FL) is particularly useful in dynamic environments where data and models evolve over time. However, this scenario presents unique challenges and opportunities. For instance, Models must adapt quickly to new data and changes in user behavior, we need to track down these changes and noises. Also, data quality and devices availability is quite difficult to foresee, therefore the system must address these details. As described by Research Questions in 1, privacy must be preserved even in an fast-moving environment.

Aside from that, federated learning comes with different opportunities and is the key-enabler technology for the following: Models can be continuously updated with new data, improving their performance over time and in real time. Also, distributing the computing load improves the overall efficiency [3]. To wrap up: FL in dynamic environments is an evolving research area with the potential to transform AI in industries such as healthcare, communications, finance, and the Internet of Things (IoT).

Projects such as [4] are similar to our proposal but does not tackle the complexity of having heterogenous environments such different architectures or models. For instance, Kim et alii in [5] employ a system which is built using high-performance commercial hardware such as Nvidia Jetson. Technologies enabled by this approach are also analyzed by Pham et alii in [6] however, the work is focused more on the enabled scaling offered by kubernetes than on the advantages or heterogeneity of the devices or models.

### 2.2. Heterogeneous Federated Learning

Using Federated learning we need to address heterogeneity in several dimensions: communication, models, statistics, and devices.

**Communication** : It can be difficult to coexist with Networks with variable bandwidth, latency, and reliability. These are difficult to manage and correct. For instance, protocols such as TCP/IP

 doi:10.20944/preprints202408.2125.v1



are compatible with changing reliability of the communication, but not so elastic in terms of infrastructural changes such IP or handover [7].

**Models** : Different model, machine-learning architectures, data formats, and data dimensions can create difficulties in aggregation. For instance, two distinct layout of neural networks can generate different results with the same data. In this case, the results need to be evaluated and analyzed with a specific algorithm [8].

**Statistics** : Non-uniform data distributions and misaligned statistics can affect the quality of the overall model. For instance dealing with heterogeneous data from different sources can be difficult and need to be adapted with specific algorithms and preparations, otherwise can lead to over-fit and under-fit problems in the models.

**Devices** : Limitations in computing power, memory, and communication capabilities between devices can slow learning process [2]. For instance, distributing the analysis between low powered end devices such as smartphones or low-end IoT devices and High End processing servers could slow down the high end devices.

Despite these challenges, FL offers opportunities to exploit data diversity and improve model robustness. Techniques such as data compression, differential federated learning, and model adaptation can mitigate heterogeneity issues. As described by Section 1 FL in heterogeneous environments is an active research field with the potential to advance distributed and adaptive artificial intelligence in complex contexts. We argue that, to the best of our knownledge, the integration of heterogenous environments and cloud-native infrastructures such as kubernetes or Docker is a novel approach.

### *2.3. Privacy Leakage Problem*

One of the crucial challenges in Federated Learning, as described by Section 1, in particularly in heterogeneous environments, is ensuring data privacy. While FL is designed to keep data decentralized and local to each participating customer, the variability and complexity of these environments introduce several channels which sensitive information can be inadvertently exposed through. This phenomenon, known as Privacy Leak, can occur through gradient reversal attacks[9], exploited model updates[10], side-channel attacks[11], and membership inference attacks[12]. Understanding and mitigating these risks is critical to the safe and effective implementation of FL, especially in sensitive industries where the diversity of devices and data sources and their marked instability and variation in the frequency of device participation in the federated learning process further complicate the privacy landscape. For this we propose the following Use Case Scenario: let us suppose to have different users, which connects to a platform trough smartphones. These smartphones can get details on the environments such as photos of environment and details on the surroundings such as road-traffic. This can be sent to a centralized server to analyze it an suggest, for instance, the best road to take to avoid traffic.

## 3. Method

### *3.1. Problem Statement and Motivation*

Federated Learning (FL) is designed to enable decentralized model training across multiple devices, ideally while preserving data privacy. However, deploying FL in dynamic and heterogeneous environments introduces several unique challenges. These environments are characterized by diverse device capabilities, varying network conditions, and non-IID (independently and identically distributed) data across devices [13].

 doi:10.20944/preprints202408.2125.v1



## 4. Statistical Methodology for Federated Learning

The use of distributed computation, enabled by techniques, described in this paper, allows for efficient resource management and optimal scalability in dynamic and heterogeneous environments. Descriptive statistics such as the mean $\mu$ and variance $\sigma^2$ are calculated locally on each node:

$$\mu_i = \frac{1}{n_i}\sum_{j=1}^{n_i} x_j, \quad \sigma_i^2 = \frac{1}{n_i - 1}\sum_{j=1}^{n_i}(x_j - \mu_i)^2 \tag{1}$$

where $x_{ij}$ represents the local data on node $i$ and $n_i$ is the number of samples. Distributed and federated computation ensure these calculations are performed in isolated and consistent environments, allowing for secure and accurate aggregation of local models. The global mean $\mu_g$ and global variance $\sigma_g^2$ can then be calculated as:

$$\mu_g = \frac{\sum_{i=1}^{k} n_i \mu_i}{\sum_{i=1}^{k} n_i}, \quad \sigma_g^2 = \frac{\sum_{i=1}^{k}(n_i - 1)\sigma_i^2}{\sum_{i=1}^{k} n_i - k} \tag{2}$$

where $k$ is the total number of participating nodes. This approach ensures robustness and flexibility in Federated Learning.

### *4.1. Federated Averaging*

The Federated Averaging (FedAvg) algorithm is a key methodology in the field of federated learning [14], a distributed learning paradigm that allows the training of machine learning models without having to centralize the data. In this approach, several devices or clients, which can be smartphones, IoT sensors, or other edge computing entities, locally train a copy of the model using their own private data. The FedAvg process works using 5 macro-points:

1. Initialization: The central server initializes the global model with initial weights and distributes it to all participating clients.
2. Local Training: Each client receives the global model and trains it locally on its data for a predefined number of epochs or iterations. This local training produces an updated set of weights specific to each client.
3. Sending Weights: Clients send their updated weights to the central server. During this process, only the model weights are transferred, while the raw data remains on local devices, thus preserving user privacy in a broad way.
4. Aggregation: The central server collects all updated weights from clients and calculates the weighted average of these weights to update the global model. Weighting can take into account the amount of local training data from each client, ensuring that clients with more data have a greater influence on updating the model.
5. Iteration: This process of deploying the global model, training locally, dispatching weights, and aggregating is repeated for a predefined number of rounds until the global model reaches a desired convergence or performance level.

This is described with more details – such as averaging formula – in Algorithm 1. The Federated Averaging algorithm's formula used is

$$\mathbf{w}_{t+1} = \sum_{k=1}^{K} \frac{n_k}{n} \mathbf{w}_{t+1}^k$$

where $\mathbf{w}_{t+1}$ is the weight vector of the updated global model; $K$ is the total number of participating devices; $n_k$ is the number of data samples on the device $k$; $n = \sum_{k=1}^{K} n_k$ is the total number of data samples across all devices; and $\mathbf{w}_{t+1}^k$ is the weight vector of the local model updated by the device $k$ per round $t + 1$.

This approach is particularly effective in heterogeneous and dynamic environments, such as those characterized by the use of IoT devices and edge computing described in this paper, as it reduces

 doi:10.20944/preprints202408.2125.v1



the need to transfer large volumes of data across the network and addresses issues related to data privacy and security. data. Furthermore, the aggregation of weights allows to obtain a robust and generalized global model, exploiting the diversity of the data distributed among clients. The privacy is not automatically maintained by default, to introduce it, we need to implement advanced techniques such as Differential Privacy, which we will highlight in details in Section 4.2.

**Algorithm 1:** FedAvg: Federated Averaging Algorithm

**Data:** Initial global model $w_0$, number of clients $K$, number of rounds $T$
**Result:** Updated global model $w_T$
**for** *each round* $t = 1, 2, \ldots, T$ **do**
  **for** *each client* $k = 1, 2, \ldots, K$ ***in parallel*** **do**
    $w_{t+1}^k \leftarrow \text{Client Update}(k, w_t)$ ; `// Client k update the model`
  $w_{t+1} \leftarrow \frac{1}{K} \sum_{k=1}^{K} w_{t+1}^k$ ; `// Weight Averaging`

### 4.2. Privacy Leakage

Attacker can exploit the privacy of the system with different tactics. Each of these tactics can lead to different results and details. This is particularly important in the field of distributed federated learning, for instance if the recent development of Recall[2], an artificial intelligence model of Microsoft, which can be used to search and retrieve images using details of their content, expressed in natural language. If systems similar to this one, uses federated machine learning and send data to a central analyzer, the privacy must absolutely be one of the core features of the system to avoid leakage of personal data of users.

**Gradient Inversion Attacks** : Attackers can reconstruct original data from the shared gradients during the model update process. Even though raw data is not directly shared, gradients can carry enough information to reveal sensitive data points. For example, given a gradient $\nabla L(\mathbf{w}, \mathbf{x})$ computed with respect to the model weights $\mathbf{w}$ and input data $\mathbf{x}$, it is possible to approximate $\mathbf{x}$ by minimizing the difference between the computed gradient and the gradient of a guessed input.

**Model Updates** : Repeatedly sharing model updates can lead to leakage of information about the training data. Over time, these updates can accumulate enough information for an attacker to infer private data. For instance, in a typical FedAvg setup, the global model update at round $t+1$ is computed as:

$$\mathbf{w}_{t+1} = \mathbf{w}_t + \eta \sum_{k=1}^{K} \frac{n_k}{n} \Delta \mathbf{w}_t^k \tag{3}$$

where $\eta$ is the learning rate, $n_k$ is the number of data points at client $k$, $n$ is the total number of data points, and $\Delta \mathbf{w}_t^k$ is the local model update from client $k$. If the model update is performed whithin a short period of time, the attacker can infer from the absence or the high volume of sent data the position or the details on the data of the user. Otherwise, if the model is not sufficiently protected in transit, the attacker can intercept it and try to analyze it.

**Side-Channel Attacks** : Attackers can exploit side-channel information, such as the timing or size of the communications between clients and the server, to gain insights into the data being processed. These side-channels can indirectly leak sensitive information even if the data and model updates are encrypted. Famous attacks such as Spectre and Meltdown fall in this category.

**Membership Inference Attacks** : These attacks aim to determine whether a specific data point was part of the training dataset. This is particularly concerning in scenarios where the presence of

[2] https://learn.microsoft.com/en-us/windows/ai/apis/recall

 doi:10.20944/preprints202408.2125.v1



certain data points can imply sensitive information. Given a model $\mathcal{M}$ and a data point $\mathbf{x}$, an attacker can train a shadow model to infer the membership status of $\mathbf{x}$ by analyzing the output confidence scores.

To mitigate these risks, several techniques have been proposed. The main and most present in literature are:

**Differential Privacy** : A technique that adds noise to the gradients or the model parameters to mask the contribution of individual data points[15]. By ensuring that the inclusion or exclusion of a single data point does not significantly affect the output, differential privacy provides a strong privacy guarantee. The formal definition of differential privacy is:

$$\Pr[\mathcal{M}(D) \in S] \leq e^{\epsilon} \Pr[\mathcal{M}(D') \in S] + \delta \tag{4}$$

where $\mathcal{M}$ is the randomized mechanism, $D$ and $D'$ are datasets differing by one element, $S$ is a subset of possible outputs, $\epsilon$ is the privacy budget, and $\delta$ is a small probability.

**Secure Multi-Party Computation (SMPC)**: SMPC protocols[16] allow multiple parties to jointly compute a function over their inputs while keeping those inputs private. In the context of FL, SMPC can be used to securely aggregate model updates without exposing individual contributions. For example, using additive secret sharing, each client $k$ splits its update $\Delta \mathbf{w}_t^k$ into shares and distributes them among the clients. The server only receives the aggregated result, which is the sum of all shares.

**Homomorphic Encryption**: This allows computations to be performed on encrypted data without needing to decrypt it first. In FL, homomorphic encryption can be used to perform model aggregation securely, ensuring that the server does not see the raw updates from clients. Given an encryption function $E$ and a decryption function $D$, homomorphic encryption ensures that:

$$D(E(a) \cdot E(b)) = a + b \tag{5}$$

Unfortunately, at the moment of writing, Homomorphic Encryption is orders of magnitude slower than non-homomorphic alternatives, making it unfeasible for large amount of data.

## 5. Application of Multilayer Architecture

To address the challenges exposed in the previous sections, we propose to use a virtual machine (VM) cloud environment with Docker and Kubernetes (specifically its implementation called k3s) containers. This approach offers a number of advantages, which maps to the following:

**Scalability** : The cloud environment can be easily scaled to meet changing needs for computational resources. By leveraging cloud platforms, resources can be dynamically allocated based on current demand, allowing for seamless expansion or contraction of the infrastructure. This is particularly useful in federated learning (FL), where the number of participating devices and the volume of data can vary significantly over time. Autoscaling features in Kubernetes ensure that container instances are automatically adjusted to handle fluctuating workloads, maintaining optimal performance without manual intervention.

**Flexibility** : Containers can be used to isolate and manage different components of the FL system, making it easy to adapt to heterogeneous devices and data. Each container can be configured with the specific dependencies and environment required for a particular task, ensuring consistency and reproducibility across different nodes. This modular approach allows for rapid deployment of updates and new features, as well as easier troubleshooting and maintenance. Furthermore, containers enable the use of diverse programming languages and tools within the same FL system, enhancing the ability to integrate with various data sources and device capabilities.

**Reliability** : The cloud environment provides a reliable and resilient infrastructure that can tolerate node failures and other issues. Cloud providers typically offer robust service level agreements

 doi:10.20944/preprints202408.2125.v1



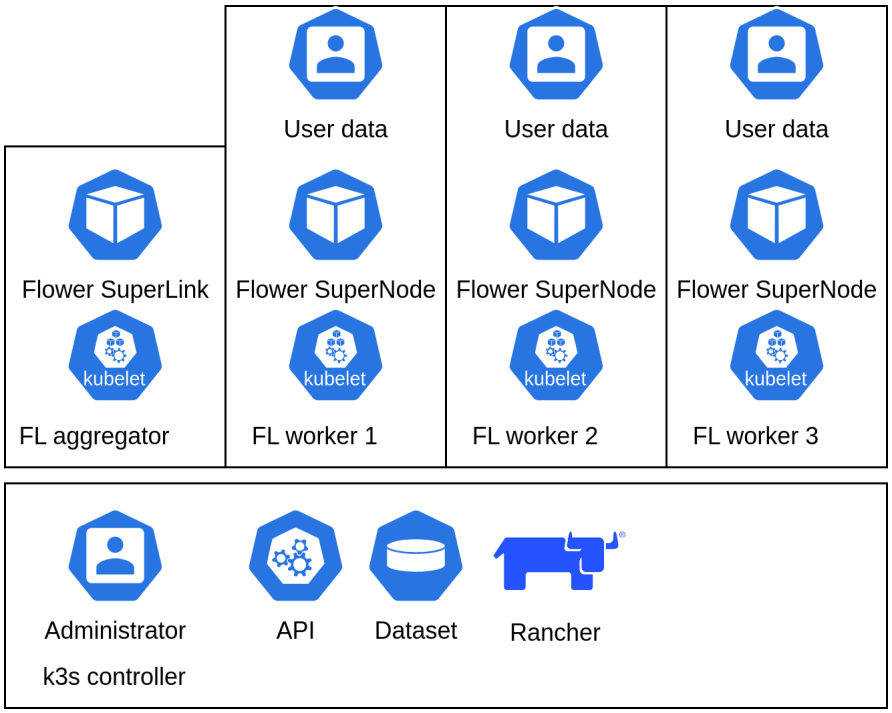



**Figure 1.** Schema of implementation using k3s. The k3s worker can be any architecture compatibile with kubernetes, with etherogeneous architecture. For instance, in our implementation, k3s worker are ARM virtual machine running in a cloud provider (Oracle)

(SLAs) and fault-tolerant architectures that ensure high availability. Docker and Kubernetes contribute to this reliability by managing container health checks, restarts, and failovers. In the event of a node failure, Kubernetes can seamlessly migrate workloads to other healthy nodes, minimizing downtime and maintaining the continuity of the FL process. Additionally, the use of multi-zone or multi-region deployments can further enhance the resilience of the system against localized failures [17].

In detail, the proposed architecture is composed of different building blocks. The main blocks are pictured in Figure 1 and are the following:

**A cluster of VMs in the cloud** : VMs provide the isolation and computational resources needed to run containers. Each VM can host multiple containers, offering a layer of abstraction that separates the hardware from the application layer. This isolation ensures that each container operates in a controlled environment, preventing conflicts and enhancing security. By distributing containers across multiple VMs, the system can leverage the cloud's elasticity to optimize resource utilization and cost efficiency.

**A container orchestrator (Docker and Kubernetes)** : The orchestrator manages the lifecycle of containers, ensuring they are always running and automatically scaling them as needed. Docker provides the containerization platform, while Kubernetes (k3s is a lightweight Kubernetes distribution designed for resource-constrained environments such as edge computing and IoT devices. It simplifies Kubernetes by reducing dependencies and the overall binary size) handles the orchestration. These tools automate the deployment, scaling, and operation of application containers across clusters of hosts. Kubernetes' advanced scheduling capabilities ensure that containers are optimally placed based on resource requirements and constraints, improving efficiency and performance.

**A federated learning module** : The FL module handles communication between participants, model aggregation, and local model updating. This module is responsible for coordinating the training process, ensuring that updates from local models are securely and accurately aggregated to form

 doi:10.20944/preprints202408.2125.v1



a global model. It manages the distribution of the global model to clients, collects local updates, and performs federated averaging or other aggregation techniques. The module also handles encryption and secure communication protocols to protect the integrity and confidentiality of the data being transmitted.

**A data management module** : The data management module pre-processes the data, distributes it to participants, and ensures data privacy. This module is crucial for handling the diverse and often sensitive nature of the data used in FL. It includes functionalities for data normalization, anonymization, and encryption to comply with privacy regulations and protect user information. The module also manages the allocation of data to ensure balanced and representative training across all participants, improving the robustness and fairness of the final model.

This approach offers a flexible, scalable, and reliable solution to address the challenges of Federated Learning in dynamic and heterogeneous environments. The use of a cloud and container environment simplifies the management of the system and adapts it to different needs. With this approach, you can leverage the benefits of Federated Learning to develop effective, privacy-preserving machine learning models in complex, ever-changing environments. The integration of advanced container orchestration and cloud capabilities not only enhances the scalability and flexibility of the system but also ensures robustness and reliability, making it well-suited for a wide range of applications where data privacy and security are priorities.

#### 5.0.1. Architectural Support using Docker and Kubernetes

With the application of this multilayer architecture in dynamic and heterogeneous environments, the use of container technologies like Docker and Kubernetes (k3s) plays a crucial role in mitigating privacy leakage:

- **Isolation and Consistency**: Containers provide isolated environments for each participating node, ensuring that computations are consistent and reproducible. This isolation limits the potential for privacy leakage between nodes.
- **Orchestration and Scaling**: Kubernetes automates the deployment, scaling, and operation of containers. This ensures efficient resource management and helps in dynamically adapting to changing computational demands, which is essential in large-scale FL systems.
- **Secure Communication**: Containers can be configured to enforce secure communication protocols, ensuring that data in transit is protected. Kubernetes can manage and automate the deployment of these secure channels.
- **Automated Updates**: Integration with Continuous Integration/Continuous Deployment (CI/CD) tools ensures that security patches and updates are consistently applied across all nodes, reducing vulnerabilities.

### *5.1. Implementation*

The system was implemented and analyzed using Oracle Cloud Infrastructure (OCI)[3]. This infrastructure offers x86-based virtual machines and ARM64-based virtual machines. We initially implemented our cluster using three ARM-based virtual machines and connected them using a private network. That is, after that we installed k3s controller and rancher in one of the machines, using the remaining two as k3s workers. To facilitate the usage of the cluster, we also instantiated rancher inside the cluster. Rancher is a kubernetes management platform, which enables the administrator to graphically analyze and use the cluster without dealing with advanced kubernetes configurations from command line. This enabled us to automatically manage the infrastructure, adding and removing nodes easily. To achieve the federated learning and to run the models we installed Flower[18] inside

[3] https://www.oracle.com/it/cloud/

 doi:10.20944/preprints202408.2125.v1



the container images. This tool, which is a framework to create federated learning models, removes the daunting task to adapt our model to the updates sent by the users. This is implemented by using two docker containers: flower-superlink, the main module that aggregates the workers and flower-supernode, the worker itself that runs the data-gathering and model update application. The implementation is pictured in Figure 1. In this picture, the containers, the infrastructure and the controller are visibile. The controller possess it initial dataset which can be distributed to the users (in production the dataset will be augmented with user personal data or private dataset). The nodes uses flower as the machine learning platform to analyze the data and aggregate the result to the flower superlink. Obviously, more pod than one per worker can be instantiated. The user can then access the superlink to get the result of the learning and the outcome of the learning process.

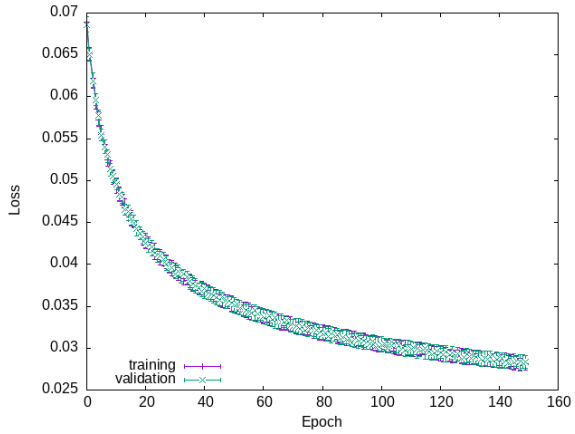
#### 5.1.1. Results

To validate our study, we analyzed the data simulating 50 containers running over three ARM virtual machines instantiated in Oracle Cloud Infrastructure. We tested our implementation with a well-known dataset such as cifar10 [4]. This dataset is non-IID distributed over the 50 machines and the results are aggregated by Flower using the super link. We run the results over 150 epochs for 10 batch runs. In Figures 2 we present accuracy and loss of our system. It is well known that accuracy in this field is still not comparable with the centralized one at the moment of writing as clearly evinctable from plots. Comparing our results with the ones present in literature [19] we clearly see a similar progression even in heterogeneous architecture, without a significant lacking in terms of performances. Our results shows a precision of 63% and a loss of 3%. This is due to the nature of federated learning, in which local minimum values are not so easy to catch by the learning process. Focusing on modern optimization such as the one described by Wang et al. in [20] it can be possible to increase the performance of the network, this improvement is still under development.

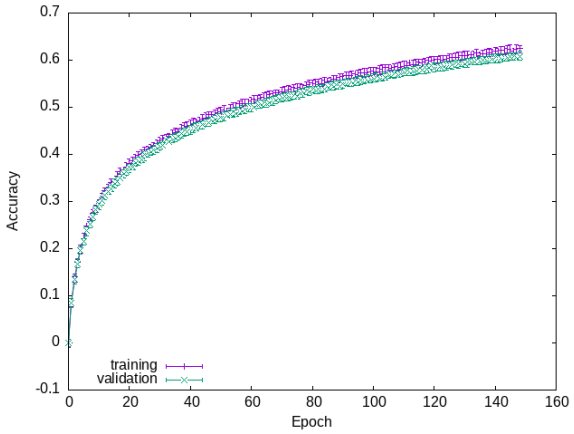
[4] An image dataset with 10 classes, available at https://www.cs.toronto.edu/~kriz/cifar.html

 doi:10.20944/preprints202408.2125.v1

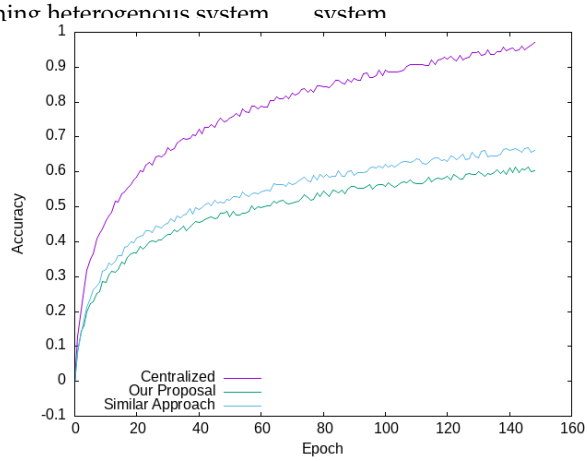





**(a)** Loss of federated learning heterogenous system




**(b)** Accuracy of federated learning heterogeneous system




**(c)** Comparison of accuracy of federated and centralized learning system and similar works.

**Figure 2.** Analisys of federated learning accuracy and loss in training and validation.

## 6. Final Considerations and Future Directions

**Advantages.** The use of containers in a federated learning environment provides numerous benefits. Containers facilitate the easy deployment and management of applications by encapsulating all necessary dependencies and configurations. This ensures consistency across different environments, reducing the chances of conflicts and errors. Furthermore, containers support rapid scaling and resource allocation, which is critical in handling dynamic workloads typical of federated learning systems. The isolation provided by containers also enhances security by limiting the potential impact of vulnerabilities within any single container.

**Disadvantages.** Despite their benefits, containers also present some challenges. One of the main disadvantages is the complexity involved in managing a large number of containers, especially in distributed environments. This requires sophisticated orchestration tools like Kubernetes, which can have a steep learning curve. Additionally, while containers improve resource utilization, they still introduce overhead compared to running applications directly on the host OS, potentially affecting performance. the security of containers is an ongoing concern, as misconfigurations or vulnerabilities can lead to breaches if not properly managed.

**Unresolved Issues.** Several unresolved issues remain in the deployment of federated learning using containers. One significant challenge is ensuring data privacy and security across distributed nodes, particularly when dealing with sensitive information. While encryption and secure communication protocols help, they do not eliminate all risks. Another issue is the heterogeneity of participating

 doi:10.20944/preprints202408.2125.v1



devices, which can vary widely in terms of computational power, connectivity, and data quality. Balancing the load and ensuring fair contribution from all nodes is an ongoing area of research. Additionally, managing the orchestration of containers in highly dynamic environments with frequent changes in node availability remains complex. Also, increasing the performance of the presented model can be challenging due to the distribution of data, leading to undesiderable low performances, compared to the centralized architecture.

**Further Developments.** Looking ahead, several developments could enhance the deployment and effectiveness of federated learning in containerized environments. One promising area is the use of edge computing, where computation is performed closer to the data source, reducing latency and bandwidth usage. Integrating advanced AI and machine learning techniques to optimize resource allocation and orchestration decisions could also improve efficiency and performance. Additionally, developing more robust security frameworks specifically designed for federated learning can address some of the current privacy and security challenges. The evolution of federated learning algorithms that can better handle heterogeneous and dynamic environments, such as those incorporating reinforcement learning for adaptive learning rates and model updates, is another crucial area. Finally, enhancing interoperability standards among different container orchestration platforms can simplify the integration of various systems and improve overall flexibility.

### *6.1. Conclusion*

In this work we presented a distributed and heterogeneous infrastructure to test, manage, and create federated learning scenarios. This approach is promising but still lacks performances and stable approaches to deal with heterogenous systems. In conclusion we reply to Research Questions presented in section 1. During the entire paper we presented briefly scenarios, implementative details and downsides of this approach, replying to Q1 and Q2. The federated learning is a promising technology and can be greatly improved to create a distributed learning platform, leveraging on smartphones and personal devices. The last question is on the privacy implications and can be replied analyzing Section 2.3. In this section we describe how the research is analyzing tools such as differential privacy to ensure privacy of the data, in transit and on the devices. For the last question (Q4), we analyzed how Kubernetes can be used to create heterogeneous models that runs with different architectures such as the ubiquitous Intel 64-bit (called AMD64 or X86_64) and the nowadays extremely popular ARM architecture. We argue that this method could accelerate the development of applications based on federated learning, even on different architectures.

**Author Contributions:** Conceptualization, Fabio Liberti; Funding acquisition, Barbara Martini; Investigation, Fabio Liberti and Davide Berardi; Methodology, Fabio Liberti; Project administration, Barbara Martini; Software, Fabio Liberti; Supervision, Barbara Martini; Validation, Davide Berardi; Visualization, Davide Berardi; Writing – original draft, Fabio Liberti; Writing – review & editing, Davide Berardi.

**Funding:** This research received no external funding.

**Data Availability Statement:** The data presented in this study are openly available in Cifar repository at https://www.cs.toronto.edu/~kriz/cifar.html.

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

## References

1. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; others. Advances and open problems in federated learning. *Foundations and trends® in machine learning* **2021**, *14*, 1–210.
2. Ye, M.; Fang, X.; Du, B.; Yuen, P.C.; Tao, D. Heterogeneous federated learning: State-of-the-art and research challenges. *ACM Computing Surveys* **2023**, *56*, 1–44.
3. Jere, S. Federated Learning in Mobile Edge Computing: An Edge-Learning Perspective for Beyond" 5G. *Signal Processing (eess.SP)* **2020**. doi:10.48550/arXiv.2007.08030.

 doi:10.20944/preprints202408.2125.v1




4. Parra-Ullauri, J.M.; Madhukumar, H.; Nicolaescu, A.C.; Zhang, X.; Bravalheri, A.; Hussain, R.; Vasilakos, X.; Nejabati, R.; Simeonidou, D. kubeFlower: A privacy-preserving framework for Kubernetes-based federated learning in cloud–edge environments. *Future Generation Computer Systems* **2024**, *157*, 558–572.
5. Kim, J.; Kim, D.; Lee, J. Design and implementation of kubernetes enabled federated learning platform. 2021 international conference on information and communication technology convergence (ICTC). IEEE, 2021, pp. 410–412.
6. Pham, K.Q.; Kim, T. Elastic Federated Learning with Kubernetes Vertical Pod Autoscaler for edge computing. *Future Generation Computer Systems* **2024**, *158*, 501–515.
7. Hansmann, W.; Frank, M. On things to happen during a TCP handover. 28th Annual IEEE International Conference on Local Computer Networks, 2003. LCN'03. Proceedings. IEEE, 2003, pp. 109–118.
8. Melin, P.; Monica, J.C.; Sanchez, D.; Castillo, O. Multiple ensemble neural network models with fuzzy response aggregation for predicting COVID-19 time series: the case of Mexico. Healthcare. MDPI, 2020, Vol. 8, p. 181.
9. Muñoz-González, L.; Biggio, B.; Demontis, A.; Paudice, A.; Wongrassamee, V.; Lupu, E.C.; Roli, F. Towards poisoning of deep learning algorithms with back-gradient optimization. Proceedings of the 10th ACM workshop on artificial intelligence and security, 2017, pp. 27–38.
10. Zhou, X.; Xu, M.; Wu, Y.; Zheng, N. Deep model poisoning attack on federated learning. *Future Internet* **2021**, *13*, 73.
11. Chabanne, H.; Danger, J.L.; Guiga, L.; Kühne, U. Side channel attacks for architecture extraction of neural networks. *CAAI Transactions on Intelligence Technology* **2021**, *6*, 3–16.
12. Hu, H.; Salcic, Z.; Sun, L.; Dobbie, G.; Yu, P.S.; Zhang, X. Membership inference attacks on machine learning: A survey. *ACM Computing Surveys (CSUR)* **2022**, *54*, 1–37.
13. Ma, X.; Zhu, J.; Lin, Z.; Chen, S.; Qin, Y. A state-of-the-art survey on solving non-IID data in Federated Learning. *Future Generation Computer Systems* **2022**, *135*, 244–258.
14. Qu, Z.; Lin, K.; Li, Z.; Zhou, J. Federated learning's blessing: Fedavg has linear speedup. ICLR 2021-Workshop on Distributed and Private Machine Learning (DPML), 2021.
15. Huang, Z.; Hu, R.; Guo, Y.; Chan-Tin, E.; Gong, Y. DP-ADMM: ADMM-based distributed learning with differential privacy. *IEEE Transactions on Information Forensics and Security* **2019**, *15*, 1002–1012.
16. Lindell, Y. Secure multiparty computation. *Communications of the ACM* **2020**, *64*, 86–96.
17. 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 Communications Surveys & Tutorials* **2023**, *25*, 2892–2950. doi:10.1109/COMST.2023.3316615.
18. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Fernandez-Marques, J.; Gao, Y.; Sani, L.; Li, K.H.; Parcollet, T.; de Gusmão, P.P.B.; others. Flower: A friendly federated learning framework **2022**.
19. Shamsian, A.; Navon, A.; Fetaya, E.; Chechik, G. Personalized federated learning using hypernetworks. International Conference on Machine Learning. PMLR, 2021, pp. 9489–9502.
20. Wang, J.; Li, Y.; Ye, R.; Li, J. High Precision Method of Federated Learning Based on Cosine Similarity and Differential Privacy. 2022 IEEE International Conferences on Internet of Things (iThings) and IEEE Green Computing & Communications (GreenCom) and IEEE Cyber, Physical & Social Computing (CPSCom) and IEEE Smart Data (SmartData) and IEEE Congress on Cybermatics (Cybermatics); IEEE: Espoo, Finland, 2022; pp. 533–540. doi:10.1109/iThings-GreenCom-CPSCom-SmartData-Cybermatics55523.2022.00105.