applied sciences

MDPI

Article

Federating Medical Deep Learning Models from Private Jupyter Notebooks to Distributed Institutions

Laëtitia Launet [1,†], Yuandou Wang [2,†], Adrián Colomer [1], Jorge Igual [3], Cristian Pulgarín-Ospina [1], Spiros Koulouzis [2,4], Riccardo Bianchi [2,4], Andrés Mosquera-Zamudio [5], Carlos Monteagudo [5], Valery Naranjo [1] and Zhiming Zhao [2,*]

1 CVBLab, Instituto Universitario de Investigación en Tecnología Centrada en el Ser Humano (HUMAN-Tech), Universitat Politècnica de València, 46022 Valencia, Spain
2 Multiscale Networked Systems, University of Amsterdam, 1098 XH Amsterdam, The Netherlands
3 Instituto de Telecomunicaciones y Aplicaciones Multimedia (ITEAM), Departamento de Comunicaciones, Universitat Politècnica de València, 46022 Valencia, Spain
4 LifeWatch ERIC, Virtual Lab. & Innovation Center (VLIC), 1098 XH Amsterdam, The Netherlands
5 Pathology Department, Hospital Clínico Universitario de Valencia, Universidad de Valencia, 46010 Valencia, Spain
* Correspondence: z.zhao@uva.nl
† These authors contributed equally to this work.

Abstract: Deep learning-based algorithms have led to tremendous progress over the last years, but they face a bottleneck as their optimal development highly relies on access to large datasets. To mitigate this limitation, cross-silo federated learning has emerged as a way to train collaborative models among multiple institutions without having to share the raw data used for model training. However, although artificial intelligence experts have the expertise to develop state-of-the-art models and actively share their code through notebook environments, implementing a federated learning system in real-world applications entails significant engineering and deployment efforts. To reduce the complexity of federation setups and bridge the gap between federated learning and notebook users, this paper introduces a solution that leverages the Jupyter environment as part of the federated learning pipeline and simplifies its automation, the Notebook Federator. The feasibility of this approach is then demonstrated with a collaborative model solving a digital pathology image analysis task in which the federated model reaches an accuracy of 0.8633 on the test set, as compared to the centralized configurations for each institution obtaining 0.7881, 0.6514, and 0.8096, respectively. As a fast and reproducible tool, the proposed solution enables the deployment of a cross-country federated environment in only a few minutes.

Keywords: federated learning; Jupyter notebook; medical image analysis; collaborative models; cloud environment; distributed medical applications

Citation: Launet, L.; Wang, Y.; Colomer, A.; Igual, J.; Pulgarín-Ospina, C.; Koulouzis, S.; Bianchi, R.; Mosquera-Zamudio, A.; Monteagudo, C.; Naranjo, V.; Zhao, Z. Federating Medical Deep Learning Models from Private Jupyter Notebooks to Distributed Institutions. *Appl. Sci.* **2023**, *13*, 919. https://doi.org/10.3390/app13020919

Academic Editor: Krzysztof Koszela

Received: 18 November 2022
Revised: 14 December 2022
Accepted: 3 January 2023
Published: 9 January 2023

1. Introduction

The advent of artificial intelligence (AI) and its application to medical imaging has opened the way to the reduction of doctors' workloads and, ultimately, improved patient care. However, despite the growing expertise in this field, the development of optimal deep learning (DL) models relies on large quantities of training data, that are too often still difficult to obtain, especially in the medical field.

Although AI developers do have the expertise needed to develop cutting-edge models with great potential for optimal automatic diagnosis systems, local learning with limited local datasets is not sufficient. To mitigate this long-lasting data hurdle, AI experts have come up with novel methods such as the generation of additional synthetic data [1] and state-of-the-art techniques for data augmentation [2]. However, synthetic data generation still raises various concerns and is not optimal yet for developing robust and reliable

medical models for clinical practice [1]. Alongside the efforts to generate additional data samples to train DL models, it is noteworthy that the substantial volumes of medical data, necessary for optimal model training, do actually exist in other institutions.

If given access to these diverse medical institutions' local data, AI experts would be able to train optimal models with substantial datasets, thus bringing the promise of unprecedented research progress in the medical field. However, those medical data are decentralized by nature due to the privacy legislation they entail, and therefore usually do not allow for their integration across institutions without raising legal concerns. Although anonymization and /or pseudonymization were long thought to overcome these limitations and thus eventually allow safe transfers of medical data, it has been shown that it is unfortunately not sufficient to promise full patient privacy [3].

As a way to guarantee data privacy concerns while gaining access to more substantial datasets, federated learning (FL) [4,5] was introduced to collaboratively train DL models over decentralized data and already proved to perform similarly to models exposed to single-institutional data during training [6]. However, despite the growing interest in implementing FL in the medical field, this paradigm is still in its infancy. Global collaboration and data sharing among organizations in the federation settings are important requests but still open issues [7]. Indeed, most of these studies were often limited to simulations because of the complex deployment of FL settings in real-world medical applications.

Although data scientists actively share their prototype code by means of notebook environments such as Jupyter, implementing an FL system requires significant engineering and deployment efforts with the development of models at each source, their aggregation, and the communication protocols for their integration. This whole process is particularly demanding, time-consuming, and even burdensome for domain-specific users, such as pathologists at hospitals, to achieve real goals of using AI at the edge, as it requires the customization of individual models and the deployment of the remote infrastructure, not to mention the debugging process to ensure its proper functioning. As a result, Jupyter users face a set of limiting challenges that clearly highlight the gap between the Jupyter environment and FL.

Prior work [8] introduced how code fragments in Jupyter Notebooks could be containerized and reused as workflow building blocks to scale up scientific experiments to cloud infrastructures through an example of processing massive light detection and ranging (LiDAR) data. However, it did not support any AI-based code implementations. In this paper, we extend our previous work and present a reproducible step-by-step guideline that allows for generating a federated set-up base starting from a centralized Jupyter code for distributed medical applications. To be specific, our main contributions include:

- Reduce the complexity of implementing a federated learning framework by making use of a Jupyter notebook code template for artificial intelligence experts;
- Improve the deployment efficiency of cross-silo federated learning systems by automatically containerizing the Jupyter components to provide to distributed users;
- Seamlessly bridge the gap between Jupyter and FL systems by providing a generic solution that is both easy-to-use and reproducible;
- Give an example tutorial of the proposed solution, the Notebook Federator, by applying it to a real-world cross-countries computational pathology classification task.

2. Related Work

Although federated learning (FL) was initially developed for mobile and edge device use cases under a cross-device paradigm [9], cross-silo FL recently gained traction in various domains, including healthcare [10]. Under this setting, models are trained on siloed data across organizations, and medical data remains local and decentralized in each collaborating institution (client) [9]. Only the resulting weights and parameters are shared and aggregated together before being transferred back to the different clients.

In recent years, these applications have become increasingly popular in the medical image analysis field, as shown in some successful implementations, including brain imaging

classification and segmentation [6,11,12], EEG signal classification [13], and, more recently, histopathology [14,15]. Li et al. [6] demonstrated the feasibility of using a client–server FL system to perform brain tumor segmentation. Roy et al. [11] introduced BrainTorrent, which is a peer-to-peer serverless FL environment, to perform a whole-brain segmentation. Lu et al. [15] presented a large-scale computational pathology study to demonstrate the feasibility and effectiveness of privacy-preserving FL using thousands of gigapixel whole slide images from multiple institutions. However, due to the complexity of FL implementation in real-world situations and large-scale deployment, most of these medical applications of FL still rely on simulations.

To support the different FL scenarios and simplify their deployment and execution process, some studies proposed easy-to-use open-source FL frameworks, such as PySyft [16], OpenFL [17], FLOWER [18] or Federated Scope [19]. The majority of these solutions focuses on the architecture of the FL framework. Whereas FLOWER [18] provides a core framework architecture with Edge Client Engine and Virtual Client Engine, Federated Scope [19] proposes an event-driven architecture with an asynchronous training protocol in FL for heterogeneity. These open-source frameworks have served as a basis to build federated setups, some of them in the medical field. In [20], Lee et al. leveraged the PySyft framework to implement a thyroid ultrasound image classifier across six medical institutions and compared the performance of federated training with that of conventional centralized model training. In the same vein, Florescu et al. [21] made use of FLOWER to simulate an FL system for COVID-19 detection on CT images, where the clients were deployed locally on a single machine. In contrast, in a work on OCT imaging [22], although the authors did mention the existence of FL frameworks, they chose to simplify their implementation through an independent development and to simulate the clients' nodes locally on a single supercomputer cluster.

Despite these available FL frameworks, there is no explicit pipeline for the FL training process between the framework and data scientists in these solutions, and most steps have to be done manually without FL automation. It is thus still time-consuming for domain-specific users to deploy their FL applications, not only because of the heterogeneous resources but also because of the complexity of models, such as software dependencies and parameters to deploy.

Additionally, although many AI researchers make use of Jupyter notebooks to develop DL models, there is a clear gap between the Jupyter environment and FL systems. Although some specific frameworks do provide a version of the FL code presented in a Jupyter notebook [19], there is still a need to upload the data to the cloud, and these notebooks are not fully integrated into the FL pipeline. In fact, they are mostly based on simulations rather than real-world scenarios running on different clients' machines.

Despite the many system construction efforts for FL frameworks, there is a clear need to simplify their deployment process and address the gap between Jupyter users and FL systems. Far from aiming to replace the available FL frameworks, in this work, we seek to fill the gap they are not covering by proposing a solution that can be built upon them. Therefore, we propose a novel FL research asset that leverages the Jupyter environment and proposes to reduce the complexity of implementing and deploying a reproducible FL pipeline, the Notebook Federator.

3. Notebook Federator

The Notebook Federator comes as a handy tool that combines resources rather than aiming to replace them. Therefore, we leverage different previously developed solutions to provide a generic solution that can adapt to the particular needs of the end users.

One of the groups participating in this study previously developed a tool that embeds a virtual research environment (VRE) into the Jupyter environment, Notebook-as-a-VRE (NaaVRE) (https://github.com/QCDIS/NaaVRE, accessed on 2 January 2023) [8], which comes with a toolbox aiming to provide researchers with a variety of functional components

(services). In this work, we make use of NaaVRE and its component containerizer service to Dockerize the components to share with distributed users automatically.

To implement the proposed solution, we selected a commonly used FL framework as a basis for the Notebook Federator, the FLOWER framework [18]. We chose this particular framework for its broad community and documentation and the possibility to easily modify and integrate it for our specific pipeline. However, as a technology, the Notebook Federator is generic, and could thus be used together with other FL frameworks.

3.1. System Definition and Requirements

For the efficient implementation of a framework that promotes the optimal collaboration of AI scientists, the setup proposed in this work is based on a user-friendly environment, Jupyter Notebook. However, Jupyter relies on the kernel (engine) for interpreting the code and is often limited to the capacity of the machine it is running on, thus making it difficult to support FL. To improve this, we identify a set of requirements. First, users registered within a federation should be able to develop the model architecture and code blocks that will be used to train all local models, and the central node should then automatically be able to update the global model's weights upon completing a communication round. Finally, the proposed pipeline needs to be reproducible and easily implementable for other federations.

3.2. Challenges and Assumptions

Implementing an FL ecosystem in a real-world medical environment comes with a myriad of challenges related to its practical application. First and foremost, not all medical institutions can count on the necessary hardware resources to train AI models. When they do, these might vary across sites, as do operating systems and network conditions. Moreover, different collaborators might store their raw data in varying formats and structures, thus limiting the automation of the data-loading process. Therefore, to enable the correct practical implementation of the FL setup, we proceed on the basis that the following assumptions are valid:

- All the participating institutions are already registered as part of the federation;
- Local data cannot be shipped outside the institution;
- Local data are stored on each client machine following the same—previously defined—nomenclature for labels and metadata across institutions. As some institutions may have their own tools or methods to process raw data, we assume the data have passed the quality check;
- Each medical collaborator either has local machines (e.g., local computers) to run the code or an agreement with a research group that acts as a local infrastructure (e.g., high-performance servers, or computer clusters) and is thus able to train models;
- Each participating entity has a Docker cluster or is able to build Docker images to run the corresponding code.

To reproduce the FL process, it requires configuring the corresponding software dependencies, input and/or output parameters, and services. In a traditional way, a user has to manually install all necessary software, enable services, and feed with suitable inputs, a process that is burdensome and time-consuming.

3.3. Architecture

The main goal of this work is to simplify the process of federation setup and make it less burdensome and time-consuming for domain-specific users such as AI experts and pathologists. In this work, we propose our FL setup solution, as depicted in Figure 1, illustrating an overview of its architecture. It comprises the following five steps:

1. **Model and aggregator definition:** Suppose a Jupyter Notebook user, e.g., an AI expert, develops a cutting-edge model architecture, aggregation function, and other machine learning-related code fragments for the FL training process.

2. **Create FL pipeline building blocks:** For flexibility purposes when it comes to handling collaborating institutions and updates, as well as reducing the complexity of FL deployment and execution, we propose to adopt one Jupyter Notebook extension named component containerizer on the local experimental environment, e.g., NaaVRE [8], to encapsulate FL pipeline building blocks as reusable services, such as model and aggregator.
3. **Build and push job automation:** Once the model and aggregator are ready, with the workflow of the GitHub project and docker registry, such components can be automatically built and pushed to the Docker Hub.
4. **Deliver FL building blocks to distributed resources:** The model can be delivered to distributed client users worldwide, i.e., geographically distributed clients in institutions A, B, and C, by pulling the model to local sites. At the same time, the aggregator can be easily delivered to the cloud infrastructure by pulling the aggregator to the cloud virtual machine.
5. **Federation setup:** Once the aggregator is assigned to the cloud infrastructure, e.g., a cloud virtual machine, it is easy to start the Docker container with specific IP and port number. For geographically distributed users, i.e., clients across wide-area institutions, the local AI model training process mainly contains (A) feeding with local data, (B) assigning suitable computation, and (C) starting the model training with specific data and computation (e.g., GPU resources) on site. By this time, the FL starts.

Specifically, the proposed solution enables a standard centralized AI environment to a decentralized (e.g., raw data) federated setup, flexibly supporting "using AI at the edge". In such a setting, only the models' general characteristics are shared. The Notebook Federator provides a Jupyter notebook template that gives the overall structure of the different functions and classes that will be further integrated into the FL pipeline code (https://github.com/QCDIS/Notebook-Federator, accessed on 2 January 2023).

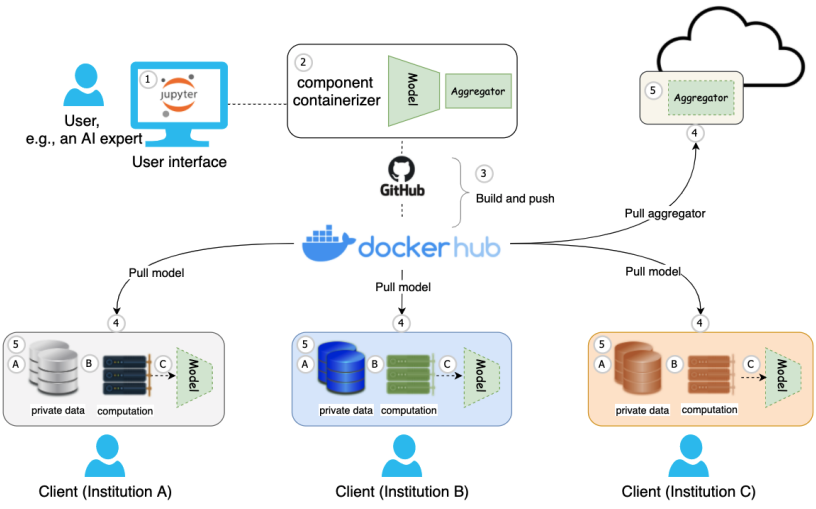

Figure 1. An overview of the architecture of Notebook Federator with NaaVRE [8].

3.4. Technology Considerations

To achieve the automation described in the architecture, we consider the following technologies.

- **Jupyter environment**—We build the FL pipeline with Jupyter Notebook. By default, we suppose that AI experts use a Jupyter environment such as Notebook to design the architecture of the model and aggregation for FL experiments.
- **Component containerizer**—We encapsulate the model and aggregator as reusable FL building blocks based on a Jupyter Notebook. The component containerizer module is one of the Jupyter extensions in NaaVRE.

- **Docker Hub**—We share model and aggregator with distributed users around the world. Docker Hub, i.e., a central repository of containers, is the easiest way to deliver reusable container applications anywhere.
- **Docker Engine**—We enhance the automation of the FL deployment and execution. In this work, we consider the Docker Engine as the critical technology for FL pipeline automation because of its popularity in the community and flourishing software tools such as Docker container, docker-nvidia, docker-compose, or even Docker Swarm. In this paper, we mainly utilize docker-nvidia for local client training with CUDA GPU resources for automating FL pipelines, as we suppose that client users have their demands for local autonomy, e.g., controlling their own data and computation for AI model training, although using Docker Swarm can also achieve large-scale automated deployment for client users.

All in all, the proposed work leverages the available technology assets presented above to bring them together for a specific application, the deployment of FL systems. With all that, the Notebook Federator positions itself as a novel module within the previously developed NaaVRE toolbox [8], providing a Jupyter notebook structured baseline for data scientists to easily reproduce the FL pipeline.

4. Case Study: Histological Image Analysis

4.1. Use Case Scenarios

We demonstrate the feasibility of the proposed pipeline on a medical image dataset of histological whole slide images (WSIs). These types of data come from the digitization of histological tissue slides into high-resolution images containing several levels of magnification, similar to those of a microscope. More precisely, the WSI dataset used in this study consists of a spitzoid melanocytic lesions dataset of 84 biopsies provided, labeled, and annotated by dermatopathologists from the Pathology Laboratory of the University Clinic Hospital of Valencia. This type of uncommon neoplasm that originates from the melanocytes is associated with ambiguous histological features and clinical behavior [23], thus representing a formidable challenge for dermatopathologists.

Due to the nature of WSIs, histological images are particularly vast (e.g., often more than $100,000 \times 100,000$ pixels) and require a lot of storage space, as a single WSI often exceeds 1GB, thus leading to heavy computational operations when training models. For these reasons, a typical method used in WSI analysis involves cutting down these images into small patches.

In this case study, we perform a patch-level region of interest (ROI) classification, that is to say, differentiate tumorous patches from non-tumorous ones. To do so, we divide the WSIs into small patches of 224×224 pixels to make the computational needs lighter and allow the use of well-known model architectures with images of the same size. Figure 2 depicts a few patch examples of the type of histological data fed into the convolutional neural network in this use case.

Hundreds to thousands of patches can be extracted from a single WSI. Table 1 details the number of extracted patches per class, i.e., tumorous vs. non-tumorous, for the WSIs available at each of the three institutions simulated in this case study. To demonstrate the potential of our method, we use twenty of the available biopsies, four of which are used for testing purposes to allow a stable comparison of results across institutions. It is important to highlight the fact that the number of patches is not correlated with the number of WSIs used, as the size of ROIs might vary depending on the WSI under study.

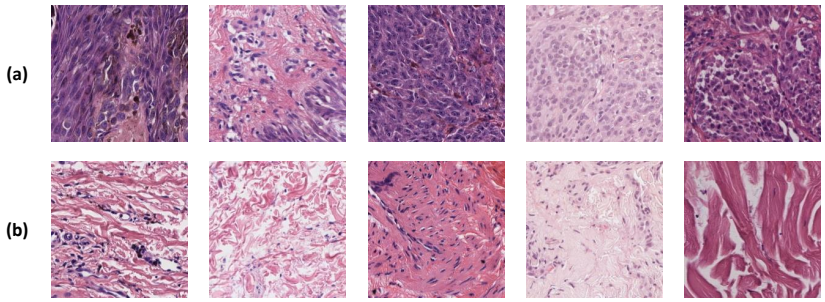

Figure 2. Examples of the type of histological patches used in this case study: (**a**) tumorous patches extracted from the annotated region made by dermatopathologists; (**b**) non-tumorous patches.

Table 1. Details of the histological image data used in this use case. Note that non-tumorous patches were then randomly undersampled to balance the dataset before training each model.

	I_A	I_B	I_C
# WSIs	10	6	8
# tumorous patches	1554	2627	1005
# non-tumorous patches	5609	4694	3979

4.2. Federation

The use case presented in this work is part of a concrete medical federation, CLARIFY (http://www.clarify-project.eu/, accessed on 2 January 2023). In this multi-sectorial and multidisciplinary consortium, nine institutions across five cities in three different countries are brought together to collaborate on the automatic analysis of specific and challenging cancer types: triple negative breast cancer, high-risk non-muscle invasive bladder cancer, and spitzoid melanocytic lesions. In particular, CLARIFY aims to develop an automated diagnostic environment for digital pathology that leverages artificial intelligence methods together with cloud computing, to enable knowledge sharing among institutions and better-informed clinical decisions.

Here, three of the nine collaborators are involved in the different steps of the proposed pipeline, both from the engineering and medical aspects. The cancer type under study in this work, spitzoid melanocytic lesions, would strongly benefit from a federation. Indeed, the few incidences of this lesion type in the population is a considerable limitation to gathering sufficient WSIs, both for DL models' training as well as for improving its clinical interpretation.

4.3. Federated Implementation

To integrate an FL pipeline in the computational pathology use case, we apply the Notebook Federator solution illustrated in Figure 1. More precisely, to carry out this implementation on this first use case scenario in realistic conditions, the model was first trained locally on private data at a single collaborating institution before being adapted and deployed for the federation by means of the Notebook Federator.

As one of the requirements of the proposed system is for it to be reproducible and easily implementable for future collaborations, we aim to give a detailed explanation of the steps undertaken to perform that federated implementation, starting from a centralized setup; in other words, the inner workings of the proposed solution. These specific steps are highlighted in Figure 3.

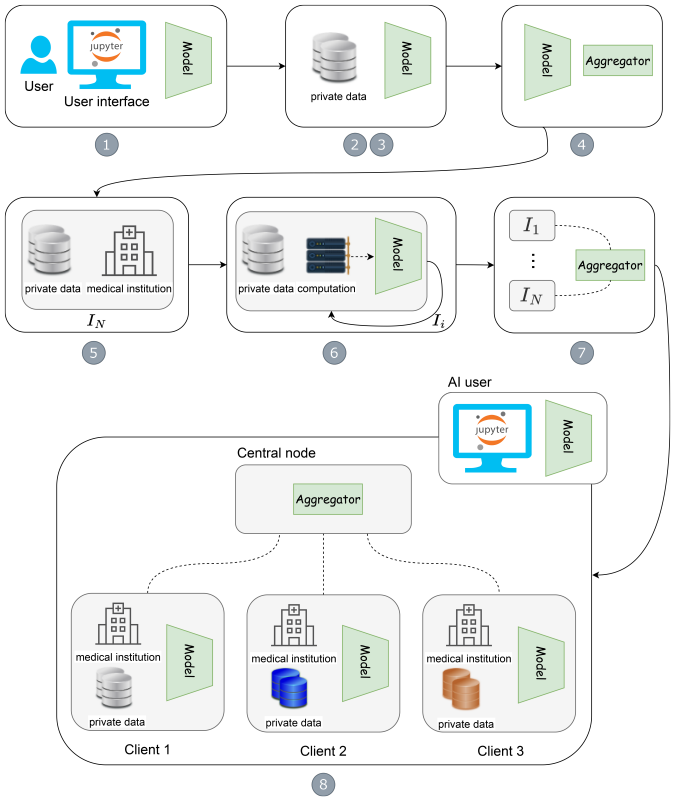

Figure 3. Step-by-step implementation of the federated pipeline, the starting point being a local institution with their corresponding local data, and the result being the federation.

First, we fill the Jupyter template provided on NaaVRE (Figure 3(1)) and define a standardized and automated data preparation protocol adapted to the specific type of data under study (Figure 3(2)). As WSI patch extraction was computed beforehand in this use case, the data loading function in the Jupyter template covers loading the patches along with their corresponding label, as well as performing an undersampling of the majority class for each set, as the dataset is clearly unbalanced (see Table 1). Then, we define a VGG16-based [24] model architecture to perform patch-level binary classification with the PyTorch 1.12 library under Python 3.6 and train it locally with the available local data at first to optimize its parameters (Figure 3(3)). To define the aggregation function to update the global model's weights when mounting the federation, we leverage the federated averaging algorithm FedAvg [25] integrated into FLOWER's strategies [18] (Figure 3(4)).

Regarding the steps involving the federation, the filled template to share with distributed users is then containerized by means of NaaVRE's component containerizer, thus allowing their further use by clients by running the corresponding images. This specific step is illustrated in Figure 4, which shows how to efficiently containerize reusable model and aggregator components with the Notebook Federator approach. The containers can thus be downloaded and set up at each of the collaborating institutions (Figure 3(5)), and model training can start (Figure 3(6),(7)). The resulting federation (Figure 3(8)) gathers the steps defined previously and communicates across nodes by means of bidirectional gRPC streams.

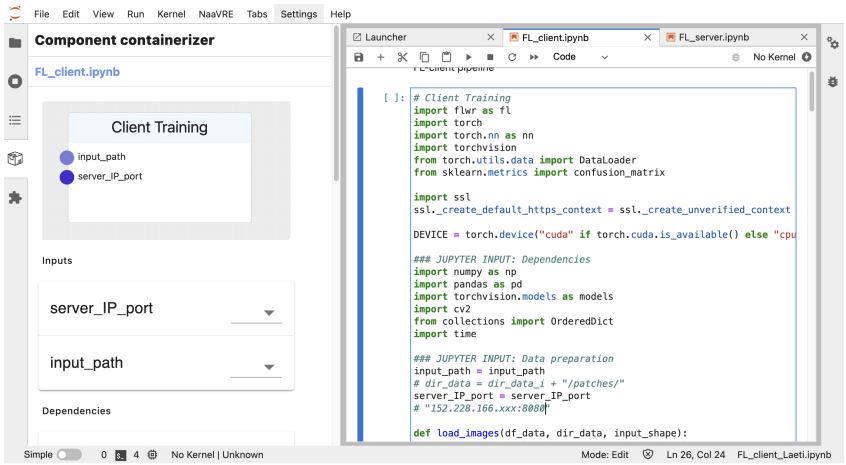

Figure 4. Demonstration of the Notebook Federator method with component containerizer in a Jupyter Notebook: building reusable FL pipeline blocks, e.g., Client Training container.

4.4. Experimental Results

4.4.1. FL System Setup

Once the code base is ready on the Jupyter Notebook user side, the component containerizer provided by NaaVRE's toolbox allows users to automate the process of building and pushing a reusable FL-based Docker container to the Docker registry in a matter of minutes. The steps to set up the concrete FL system are quick and simple, as show the times obtained for each final phase of this case study:

- Automated build–push job for the server-aggregation image container (approx. 1.9 GB): 4 m 57 s;
- Building and pushing client-training image container to the Docker hub (approx. 3.16 GB): 10 m 48 s.

Note that the processing time for the build–push jobs depends on the local network environment and image size. The server-aggregation container associated to the host IP address is then deployed on a cloud virtual machine, and distributed users can pull the client-training image and run it locally (e.g., GPU resources in Spain, the Netherlands, and even Norway), as well as mount local medical data volumes to feed the AI model training. For the communication, each client can then join the federation's training process by specifying the server IP address and port number when running the corresponding Docker containers.

4.4.2. AI Training

As depicted in Figure 5, each on-site client used different hardware resources for model training. As a result, whereas the client at institution A took around 60 seconds for 10 epochs on average, clients at institutions B and C trained each communication round for 540 and 534 seconds, respectively. All in all, the whole federation training took 46 minutes for the three communication rounds in this use case, as the central node waits for all three clients to finish training.

To allow for an effective comparison of the prediction performance of models presented in this case study, several metrics of relevance were selected: sensitivity, specificity, positive predictive value, negative predictive value, F1-score, and accuracy. A particularly important metric in pathology is sensitivity, which measures the ability of the model to correctly detect positive instances.

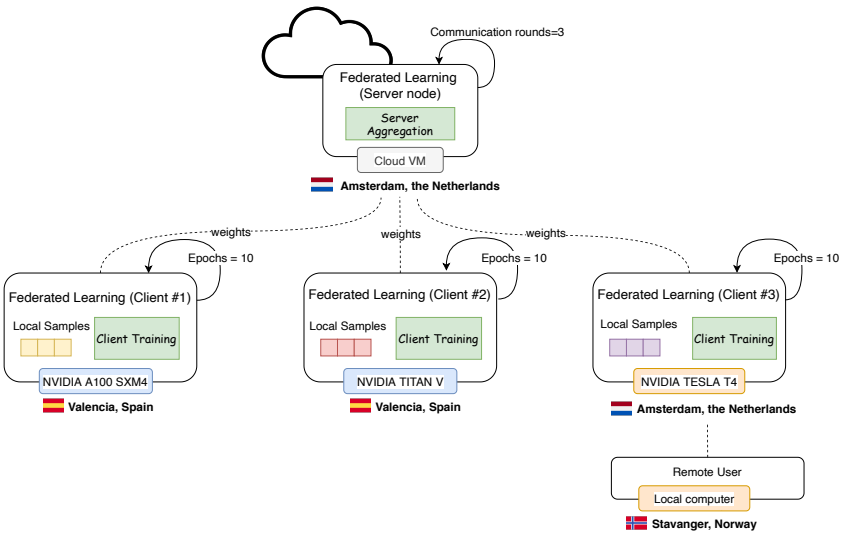

Figure 5. Implementation of the proposed solution, Notebook Federator, in a federation composed of institutions located in different countries and relying on different hardware resources.

Table 2 shows the metrics on the test set after training for 30 epochs at each institution with its own local data under a centralized configuration, compared to the results when training as a federation with all three clients for three communication rounds of ten epochs each. It is noteworthy to highlight that the scope of this study is not to reach unprecedented classification results but rather to demonstrate the federation's improvement compared to that of individual institutions. In that sense, classification metrics on the test set clearly show an overall improvement of the model trained collaboratively (federation) in relation to those trained in a centralized manner. Although I_B reaches considerably high specificity and positive predictive value metrics on the test set when trained under a centralized setup, it is important to highlight that it is mostly due to its tendency to predict a sample as positive (i.e., tumorous) even when negative, as depicted in the negative predictive value results for that model.

Table 2. Classification results reached on the test set for patch-level ROI selection on the spitzoid melanocytic lesions dataset; I_A, I_B, and I_C: centralized setups at each local institution; Federation: decentralized federated setup, leveraging all institutions' data.

	I_A	I_B	I_C	Federation
Sensitivity	0.7921	0.5896	0.7887	**0.8174**
Specificity	0.7842	**0.9874**	0.8337	0.9247
Positive predictive value	0.7813	**0.9961**	0.8457	0.9355
Negative predictive value	**0.7949**	0.3066	0.7734	0.7910
F1-Score	0.7866	0.7407	0.8162	**0.8725**
Accuracy	0.7881	0.6514	0.8096	**0.8633**

By and large, the classification results show the potential of the approach, taking into account that, the more institutions participating in model training, the more robust the resulting global model will be when exposed to unseen data.

5. Discussion

5.1. Achievements

In the presented use case, we demonstrate the feasibility of the proposed pipeline to go from a centralized training setup to a federated environment to train collaborative models. Specifically, the Notebook Federator proposes to considerably reduce the com-

plexity of FL systems' setup for AI Jupyter users by leveraging the latest technologies and available assets.

Starting from a Jupyter environment on the end-user side (i.e., the AI expert or data scientist), we put an emphasis on the reproducibility and scalability of the method for researchers. The Jupyter user only has to complete the indicated lines of the template provided as a code base, and that code is then included in the client code. Building and running the resulting Docker is fast and automatic, thus simplifying the quick implementation of the code on the client machines.

In contrast to other FL solutions and implementations, in this work, we wish to provide a real-world use case, with the server and clients located in different cities and countries, which all have access to different computing resources. Additionally, as our solution aims to simplify the complexity of FL setups and seeks to be easily reproducible, the provided use case leverages a commonly used baseline framework with a strong community and documentation, FLOWER [18]. Although the example we provide is applied to a domain-specific task in computational pathology, it is important to note that the Notebook Federator, as a technology, is generic and can easily be adapted to other tasks and baseline frameworks.

5.2. Weaknesses and Future Work

Some FL-related challenges [5] still need to be tackled in future lines of work. In this study, the GPU and CPU resources used to train the model on the different clients' machines were sufficient, but the computing resources may be heterogeneous across institutions within a federation, and some software or systems may be incompatible if the computing platform cannot handle Docker.

A typical challenge of FL approaches is client reliability, that is to say, whether a client will fail or drop out during a round. In this case study, we consider that the server can only complete a communication round when the three participating institutions finished training, and thus waits for all the clients to have completed data loading and training before performing the aggregation. Even if one of the clients ran the model training on a CPU, the central node waited to receive the weights from all three clients to perform the aggregation. In further implementations involving more clients, it could be considered to have a certain percentage of clients only needed to complete a communication round.

Another challenge brought by FL ecosystems is security. Although FL helps to tackle the data challenge in deep learning and the related major privacy limitation, it is still vulnerable to a wide range of challenging security issues. For instance, despite the anonymization process applied to medical data that was long considered sufficient to protect leaks from patient data when developing AI models, it was shown that, in some cases, sensitive information can actually be inferred [3]. Future lines will thus consider differential privacy preservation.

Currently, some security technologies can be adopted to achieve a privacy-preserving model aggregation for FL [26], such as differential privacy, multi-party computation, or even blockchain-based solutions. Additionally, Docker images may not be optimally secure because of the root access they provide to the system they are running on (https://researchcomputing.princeton.edu/support/knowledge-base/singularity, accessed on 2 January 2023). An alternative to the containerization toolkit would be singularity. This method provides the considerable advantage of limiting users' access and capabilities and includes a compatibility layer that allows Docker images to be run on the platform. The images created via our tool could therefore also be used in a singularity-enabled environment.

6. Conclusions

In this work, we propose a federated learning pipeline that allows artificial intelligence experts and medical institutions to join forces in the development of collaborative deep learning models. The proposed solution, the Notebook Federator, demonstrates the promising results of an automated federated learning framework built from Jupyter and Docker ecosystems to simplify its use by bridging the gap between artificial intelligence

developers and the Jupyter environment, as well as reducing the complexity of FL systems' deployment. As this first implementation started from an initial scenario based on several assumptions, our future lines of work will focus on deploying this framework to all collaborators of the CLARIFY medical federation presented in this paper while prioritizing the security aspect by the inclusion of differential privacy and blockchain methods into the workflow.

Author Contributions: Conceptualization, L.L., Y.W., A.C., V.N. and Z.Z.; methodology, L.L., Y.W., C.P.-O. and Z.Z.; medical data recollection and preparation, A.M.-Z. and C.M.; medical data annotation, A.M.-Z.; model preparation and data curation, L.L.; software, L.L. and Y.W.; system, Y.W.; Jupyter notebook-based toolbox updates and maintenance, S.K. and R.B.; visualization: L.L. and Y.W.; writing—original draft preparation, L.L. and Y.W.; writing—medical review and editing, C.M. and A.M.-Z.; writing—review and editing, Z.Z., V.N., A.C. and J.I.; supervision, Z.Z., V.N. and A.C.; project administration: Z.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work has been partially funded by the European Union's Horizon 2020 research and innovation programme with the project CLARIFY under Marie Sklodowska-Curie (860627), ENVRI-FAIR (824068), BlueCloud (862409), and ARTICONF (825134). This work is also supported by LifeWatch ERIC, GVA through projects PROMETEO/2019/109 and INNEST/2021/321 (SAMUEL), and the Spanish Ministry of Economy and Competitiveness through project PID2019-105142RB-C21 (AI4SKIN). The work of Adrián Colomer has been supported by the ValgrAI – Valencian Graduate School and Research Network for Artificial Intelligence & Generalitat Valenciana and Universitat Politècnica de València (PAID-PD-22).

Institutional Review Board Statement: The spitzoid melanocytic lesions dataset was collected at the Pathology Laboratory of the University Clinic Hospital of Valencia (Valencia, Spain). This study was carried out in accordance with the Declaration of Helsinki, and approved by the hospital's Ethics Committee under the approval number 2020/114.

Informed Consent Statement: Informed consent was obtained from all subjects involved in the study.

Acknowledgments: We gratefully acknowledge the support from the Generalitat Valenciana (GVA) with the donation of the DGX A100 used for this work, action co-financed by the European Union through the Operational Program of the European Regional Development Fund of the Comunitat Valenciana 2014–2020 (IDIFEDER/2020/030).

Conflicts of Interest: The authors declare no conflict of interest. The funders had no role in the design of the study; in the collection, analyses, or interpretation of data; in the writing of the manuscript; or in the decision to publish the results.

Abbreviations

The following abbreviations are used in this manuscript:

ACC	Accuracy
AI	Artificial Intelligence
DOAJ	Directory of Open Access Journals
F1S	F1-score
FL	Federated Learning
MDPI	Multidisciplinary Digital Publishing Institute
NaaVRE	Notebook-as-a-VRE
NPV	Negative Predictive Value
PPV	Positive Predictive Value
ROI	Region of Interest
SN	Sensitivity
SPC	Specificity
VRE	Virtual Research Environment
WSI	Whole Slide Image

References

1. Chen, R.J.; Lu, M.Y.; Chen, T.Y.; Williamson, D.F.; Mahmood, F. Synthetic data in machine learning for medicine and healthcare. *Nat. Biomed. Eng.* **2021**, *5*, 493–497. https://doi.org/10.1038/s41551-021-00751-8.
2. Oza, P.; Sharma, P.; Patel, S.; Adedoyin, F.; Bruno, A. Image Augmentation Techniques for Mammogram Analysis. *J. Imaging* **2022**, *8*, 141.
3. Rocher, L.; Hendrickx, J.M.; de Montjoye, Y.A. Estimating the success of re-identifications in incomplete datasets using generative models. *Nat. Commun.* **2019**, *10*, 1–9. https://doi.org/10.1038/s41467-019-10933-3.
4. Konecný, J.; McMahan, H.B.; Ramage, D.; Richtárik, P. Federated Optimization: Distributed Machine Learning for On-Device Intelligence. *arXiv* **2016**, arXiv:abs/1610.02527.
5. Li, T.; Sahu, A.K.; Talwalkar, A.; Smith, V. Federated Learning: Challenges, Methods, and Future Directions. *IEEE Signal Process. Mag.* **2020**, *37*, 50–60. https://doi.org/10.1109/MSP.2020.2975749.
6. Li, W.; Milletarì, F.; Xu, D.; Rieke, N.; Hancox, J.; Zhu, W.; Baust, M.; Cheng, Y.; Ourselin, S.; Cardoso, M.J.; et al. Privacy-preserving federated brain tumour segmentation. In *Proceedings of the International Workshop on Machine Learning in Medical Imaging*; Springer International Publishing: Cham, Switzerland, 2019; pp. 133–141.
7. Sheller, M.J.; Edwards, B.; Reina, G.A.; Martin, J.; Pati, S.; Kotrotsou, A.; Milchenko, M.; Xu, W.; Marcus, D.; Colen, R.R.; et al. Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data. *Sci. Rep.* **2020**, *10*, 1–12.
8. Zhao, Z.; Koulouzis, S.; Bianchi, R.; Farshidi, S.; Shi, Z.; Xin, R.; Wang, Y.; Li, N.; Shi, Y.; Timmermans, J.; et al. Notebook-as-a-VRE (NaaVRE): From private notebooks to a collaborative cloud virtual research environment. *Softw. Pract. Exp.* **2022**, *52*, 1947–1966. https://doi.org/10.1002/SPE.3098.
9. Kairouz, P.; McMahan, H.B.; Avent, B.; Bellet, A.; Bennis, M.; Bhagoji, A.N.; Bonawitz, K.; Charles, Z.; Cormode, G.; Cummings, R.; et al. Advances and open problems in federated learning. *Found. Trends® Mach. Learn.* **2021**, *14*, 1–210.
10. Rieke, N.; Hancox, J.; Li, W.; Milletari, F.; Roth, H.R.; Albarqouni, S.; Bakas, S.; Galtier, M.N.; L.; man, B.A.; Maier-Hein, K.; et al. The future of digital health with federated learning. *NPJ Digit. Med.* **2020**, *3*, 1–7.
11. Roy, A.G.; Siddiqui, S.; Pölsterl, S.; Navab, N.; Wachinger, C. BrainTorrent: A Peer-to-Peer Environment for Decentralized Federated Learning. *arXiv* **2019**, arXiv:abs/1905.06731.
12. Li, X.; Gu, Y.; Dvornek, N.; Staib, L.H.; Ventola, P.; Duncan, J.S. Multi-site fMRI analysis using privacy-preserving federated learning and domain adaptation: ABIDE results. *Med. Image Anal.* **2020**, *65*, 101765. https://doi.org/10.1016/j.media.2020.101765.
13. Ju, C.; Gao, D.; Mane, R.; Tan, B.; Liu, Y.; Guan, C. Federated Transfer Learning for EEG Signal Classification. In Proceedings of the Annual International Conference of the IEEE Engineering in Medicine and Biology Society, Montreal, QC, Canada, 20–24 July 2020. https://doi.org/10.1109/EMBC44109.2020.9175344.
14. Andreux, M.; du Terrail, J.O.; Beguier, C.; Tramel, E.W. Siloed Federated Learning for Multi-centric Histopathology Datasets. In *Proceedings of the Lecture Notes in Computer Science (Including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*; Springer International Publishing: Cham, Switzerland, 2020. https://doi.org/10.1007/978-3-030-60548-3_13.
15. Lu, M.Y.; Chen, R.J.; Kong, D.; Lipkova, J.; Singh, R.; Williamson, D.F.; Chen, T.Y.; Mahmood, F. Federated learning for computational pathology on gigapixel whole slide images. *Med. Image Anal.* **2022**, *76*, 102298. https://doi.org/10.1016/j.media.2021.102298.
16. Ziller, A.; Trask, A.; Lopardo, A.; Szymkow, B.; Wagner, B.; Bluemke, E.; Nounahon, J.M.; Passerat-Palmbach, J.; Prakash, K.; Rose, N.; et al., PySyft: A Library for Easy Federated Learning. In *Federated Learning Systems: Towards Next-Generation AI*; ur Rehman, M.H., Gaber, M.M., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 111–139. https://doi.org/10.1007/978-3-030-70604-3_5.
17. Reina, G.A.; Gruzdev, A.; Foley, P.; Perepelkina, O.; Sharma, M.; Davidyuk, I.; Trushkin, I.; Radionov, M.; Mokrov, A.; Agapov, D.; et al. OpenFL: An open-source framework for Federated Learning. *arXiv* **2021**, arXiv:2105.06413. https://doi.org/10.48550/ARXIV.2105.06413.
18. Beutel, D.J.; Topal, T.; Mathur, A.; Qiu, X.; Parcollet, T.; Lane, N.D. Flower: A Friendly Federated Learning Research Framework. *arXiv* **2020**, arXiv:2007.14390.
19. Xie, Y.; Wang, Z.; Chen, D.; Gao, D.; Yao, L.; Kuang, W.; Li, Y.; Ding, B.; Zhou, J. FederatedScope: A Flexible Federated Learning Platform for Heterogeneity. *arXiv* **2022**, arXiv:2204.05011.
20. Lee, H.; Chai, Y.J.; Joo, H.; Lee, K.; Hwang, J.Y.; Kim, S.M.; Kim, K.; Nam, I.C.; Choi, J.Y.; Yu, H.W.; et al. Federated learning for thyroid ultrasound image analysis to protect personal information: Validation study in a real health care environment. *JMIR Med. Inform.* **2021**, *9*, e25869.
21. Florescu, L.M.; Streba, C.T.; Şerbănescu, M.S.; Mămuleanu, M.; Florescu, D.N.; Teică, R.V.; Nica, R.E.; Gheonea, I.A. Federated Learning Approach with Pre-Trained Deep Learning Models for COVID-19 Detection from Unsegmented CT images. *Life* **2022**, *12*, 958.
22. Lo, J.; Timothy, T.Y.; Ma, D.; Zang, P.; Owen, J.P.; Zhang, Q.; Wang, R.K.; Beg, M.F.; Lee, A.Y.; Jia, Y.; et al. Federated learning for microvasculature segmentation and diabetic retinopathy classification of OCT data. *Ophthalmol. Sci.* **2021**, *1*, 100069.
23. Lodha, S.; Saggar, S.; Celebi, J.T.; Silvers, D.N. Discordance in the histopathologic diagnosis of difficult melanocytic neoplasms in the clinical setting. *J. Cutan. Pathol.* **2008**, *35*, 349–352. https://doi.org/10.1111/j.1600-0560.2007.00970.x.

24. Simonyan, K.; Zisserman, A. Very deep convolutional networks for large-scale image recognition. In Proceedings of the 3rd International Conference on Learning Representations, ICLR 2015—Conference Track Proceedings, San Diego, CA, USA, 7–9 May 2015.
25. Brendan McMahan, H.; Moore, E.; Ramage, D.; Hampson, S.; Agüera y Arcas, B. Communication-efficient learning of deep networks from decentralized data. In Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS 2017), Fort Lauderdale, FL, USA, 20–22 April 2017.
26. Kanagavelu, R.; Li, Z.; Samsudin, J.; Yang, Y.; Yang, F.; Goh, R.S.; Cheah, M.; Wiwatphonthana, P.; Akkarajitsakul, K.; Wang, S. Two-Phase Multi-Party Computation Enabled Privacy-Preserving Federated Learning. In Proceedings of the 20th IEEE/ACM International Symposium on Cluster, Cloud and Internet Computing (CCGRID 2020), Melbourne, VIC, Australia, 11–14 May 2020; pp. 410–419. https://doi.org/10.1109/CCGrid49817.2020.00-52.