Received December 24, 2021, accepted January 6, 2022, date of publication January 11, 2022, date of current version January 25, 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3141913

Decentralized Federated Learning for Healthcare Networks: A Case Study on Tumor Segmentation

BERNARDO CAMAJORI TEDESCHINI[1], (Graduate Student Member, IEEE), STEFANO SAVAZZI[2], (Member, IEEE), ROMAN STOKLASA[3], LUCA BARBIERI[1], (Graduate Student Member, IEEE), IOANNIS STATHOPOULOS[3,4], MONICA NICOLI[5], (Member, IEEE), AND LUIGI SERIO[3]

[1]Dipartimento di Elettronica, Informazione e Bioingegneria, Politecnico di Milano, 20133 Milano, Italy
[2]Institute of Electronics, Information Engineering and Telecommunication (IEIIT), Consiglio Nazionale delle Ricerche, 20133 Milano, Italy
[3]European Organization for Nuclear Physics (CERN), Technology Department, 1211 Geneve, Switzerland
[4]2nd Department of Radiology, Medical School of Athens, Attiko University Hospital, 12462 Chaidari, Greece
[5]Department of Management, Economics and Industrial Engineering, Politecnico di Milano, 20156 Milano, Italy

Corresponding author: Bernardo Camajori Tedeschini (bernardo.camajori@polimi.it)

This work was supported in part by the European Organization for Nuclear Physics (CERN) Budget for Knowledge Transfer for the Benefit of Medical Applications.

ABSTRACT Smart healthcare relies on artificial intelligence (AI) functions for learning and analysis of patient data. Since large and diverse datasets for training of Machine Learning (ML) models can rarely be found in individual medical centers, classical centralized AI requires moving privacy-sensitive data from medical institutions to data centers that process the fused information. Training on data centers thus requires higher communication resource/energy demands while violating privacy. This is considered today as a significant bottleneck in pursuing scientific collaboration across trans-national clinical medical research centers. Recently, federated learning (FL) has emerged as a distributed AI approach that enables the cooperative training of ML models, without the need of sharing patient data. This paper dives into the analysis of different FL methods and proposes a real-time distributed networking framework based on the Message Queuing Telemetry Transport (MQTT) protocol. In particular, we design a number of solutions for ML over networks, based on FL tools relying on a parameter server (PS) and fully decentralized paradigms driven by consensus methods. The proposed approach is validated in the context of brain tumor segmentation, using a modified version of the popular U-NET model with representative clinical datasets obtained from the daily clinical workflow. The FL process is implemented on multiple physically separated machines located in different countries and communicating over the Internet. The real-time test-bed is used to obtain measurements of training accuracy vs. latency trade-offs, and to highlight key operational conditions that affect the performance in real deployments.

INDEX TERMS Federated learning, learning over networks, medical imaging, healthcare networks, network architectures, machine learning.

I. INTRODUCTION

Deep learning (DL) and Artificial Intelligence (AI) have great potential in clinical research as a means for integrating complex imaging data into personalized indices of diagnosis and prognosis. Combined with the human pathologist's inputs, AI systems have contributed to significantly reduce the human error rate [1]. On the other hand, the increasing volume of data, the widespread adoption of Internet-of-Medical-Things (IoMT) [2] and AI-enabled devices with high computing capabilities, have made conventional centralized (Big-Data) learning solutions inefficient in terms of latency and scalability due to the need of moving, often periodically, large datasets. Besides, regulatory authorities as well as patient organizations are proposing stringent limitations to AI-driven data processing, to ensure that private data are not shared or transferred to third parties, even in anonymized format [3]. In such a dynamic context, Federated Learning (FL) technology [4] has been emerging as a viable solution [5]–[7]. The technology enables the distributed training of Machine Learning (ML) models over remote devices, namely the Medical Nodes (MNs), or clients,

The associate editor coordinating the review of this manuscript and approving it for publication was Larbi Boubchir.

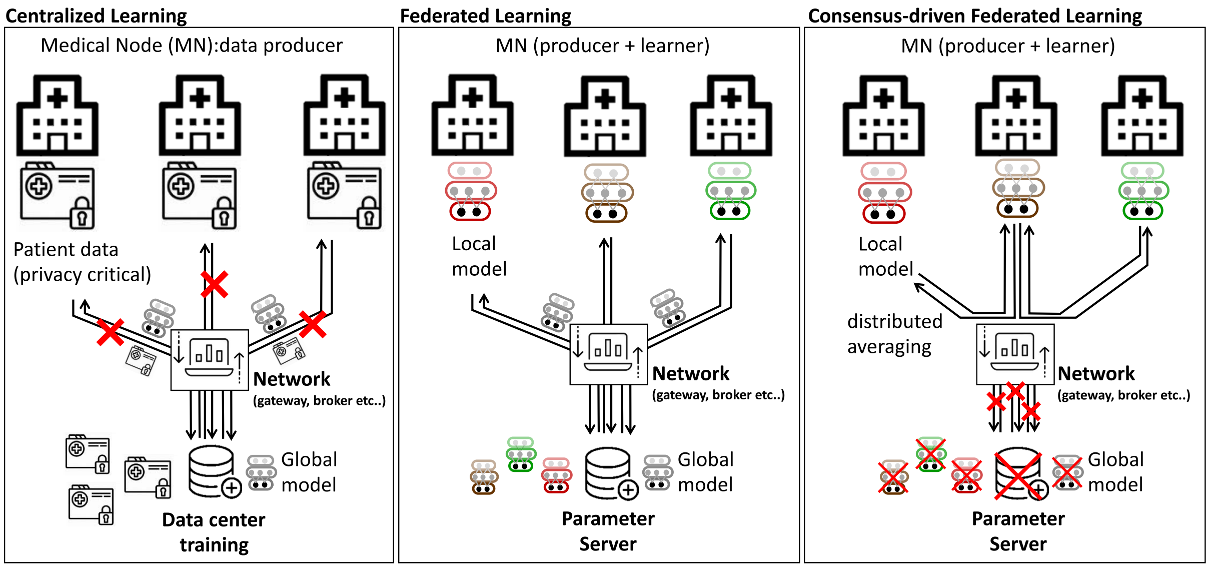


FIGURE 1. From left to right: Centralized Learning (CL), Federated Learning (FL) coordinated by the Parameter Server (PS), namely Federated Averaging (FA), and Consensus-driven learning with fully decentralized implementation (i.e., without PS).

without requiring the same devices to disclose their training data, possible containing privacy sensitive information about patients.

As shown in Figure 1, vanilla FL algorithms, such as Federated Averaging (FA) [4], [8], allow the MNs to learn a shared ML model under the orchestration of a Parameter Server (PS). Typically, the PS interacts with the medical devices through a network, i.e., using a provider or gateway, to collect (and store) the received local models. These are aggregated to obtain a global model that is then fed back to the edge devices for validation and inference. Each Medical Node (MN) thus participates in training the shared model using its own dataset. However, in contrast to classical Centralized Learning (CL), privacy-sensitive data are kept on the device, while cooperation is based on local model exchange.

FL uses the data as and when they are received or available at the MN and it thus supports flexible training processes, such as continual and incremental learning. First implementations of FL leveraged on a server-client architecture [6], [7] where the PS coordinates the learning process. On the other hand, these classical FL techniques are often considered not always resilient against model inversion attacks on the PS, where privacy-critical data can be recreated using the local models stored by the PS [9], [10].

A. RELATED WORKS

Different FL implementations have emerged in the past few years [11] targeting several application scenarios [12]–[16] and technology enablers [17]–[19]. Focusing on the popular brain tumor image segmentation challenge [20], first steps towards the integration of a privacy-preserving FL system into a medical image analysis framework are in [6], [13]. It was demonstrated that the FL model quality is comparable to that of a model trained using CL on a data fusion center. The dataset therein used to build and test the AI system is the multimodal Brain Tumor Segmentation (BraTS) set [21]–[23]. The FL process typically utilizes the same training pipeline designed for centralized training: current state of the art models for 3D brain Magnetic resonance imaging (MRI) processing are based on an auto-encoder regularization tool [24], or the U-NET model [25], [26] with hyperparameter structure described in [27].

The above approaches have two main limits. First, they only used datasets prepared ad-hoc for testing; they did not consider how real data coming from hospitals affect the training process or how to generalize the model. Second, they rely on a central fusion center for model aggregation which could lead to privacy leaks. Decentralized training on incomplete and heterogeneous image datasets (i.e., different scan modalities) poses new challenges to FL and is the main focus of this paper. For what concerns the algorithms, the training process can be implemented via vanilla FL tools that rely on the PS for distributed coordination. However, trusted PS designs are needed [28]. As an alternative, fully distributed learning tools have been recently proposed to replace, or minimize the use of the PS functions, enabling server-less training. These techniques have roots in consensus [15] and distributed ledger [29] enablers, as they let the local models be consensually shared and synchronized across multiple MNs. They rely solely on in-network processing, via consensus, diffusion [18], [30], [31] or gossip [32] tools. Fully decentralized FL policies have been considered for training on low-power devices (robots, drones) in several industrial verticals [15], [33] such as robotics, connected automated vehicles [14], [34] and medical diagnosis [35]. Network scalability/connectivity aspects are however not considered or discussed.

Though FL approaches are promising, they are often simulated on virtual frameworks [11] where (virtual) clients act as independent threads and run on the same physical machine. With the exception of [12], [36], [37], pilot demonstration of FL platforms featuring geographically distributed devices and real-time training over the Internet are currently overlooked. In line with the road-map towards native (in-network) AI designs [38], in this paper we propose a real-time platform to support network and federated learning functions integration, validating the proposed FL solution in a real-world deployment.

B. CONTRIBUTIONS

The paper proposes the application of decentralized FL methods in the context of cancer diagnosis, focusing in particular on brain tumor segmentation. To demonstrate the system in a real environment, a novel Message Queuing Telemetry Transport (MQTT) based architecture has been developed. The platform is employed to verify the performance of FL over real geographical distributed MNs characterized by non-uniform computing capabilities and heterogeneous datasets. Both classical FL based on PS designs and fully decentralized architectures are evaluated, discussing for each case their impact on the MQTT publishing/subscription operations. The proposed networking architecture and tools are designed to optimize the FL process, weaving together synchronous and asynchronous operations, as well as taking into account the training time of the individual clients to avoid performance penalties caused by slower MNs, namely the straggler effect [11], [39]. The algorithms and MQTT real-time network have been demonstrated by combining for the first time, in a decentralized FL approach, public (BraTS) and private clinical data obtained from the clinical workflow (with no pre-filtering).

The main contributions of the paper are further summarized as follows:

- Vanilla and fully decentralized FL algorithms are integrated into a novel network architecture that adopts the MQTT transport protocol to orchestrate the deep ML model parameters exchange. We propose an optimized set of information to be embedded into the MQTT payload and to characterize the real-time learning process on each epoch, discussing also model parameters compression, serialization and Quality-of-Service (QoS) mechanisms.
- Implementing FL tools on top of the MQTT protocol brings novel challenges that are discussed here for the first time. In particular, 4 mechanisms for ML parameter exchange are proposed: these account for synchronous and asynchronous operations on the MNs and the PS, respectively, as well as decentralized FL, where the clients, rather than the PS, self-organize to coordinate the FL process. For all the considered cases, the MQTT broker is configured to support the client authentication, authorization as well as to control the access to FL resources (global/aggregated models, training statistics and timing). All the proposed architectures are compared to quantify the latency/model quality trade-offs for synchronous and asynchronous FL processes.
- Validation of the FL tools is based on a federation of 5 MNs distributed in different institutions across the Europe and communicating over the Internet. Focusing on brain tumor segmentation as case study, the experiments are conducted in real clinical settings and with medical MRI images obtained from the daily clinical workflow. The proposed real-time testbed thus provides a unique opportunity to quantify the improvements of the FL process in terms of practical metrics, namely the Dice Similarity Coefficient (DSC), and on heterogeneous datasets without any pre-filtering and not prepared for testing purposes (in contrast to public and widely available data).

The paper is organized as follows. Section II introduces the FL algorithms, namely vanilla and fully decentralized tools based on consensus and analyze them with respect to medical imaging problems and privacy considerations. Section III discusses the proposed brain tumor segmentation tasks and the necessary adaptations for FL system deployment. Section IV describes the specifications of the proposed networking architecture and MQTT protocol integrated designs. Targeting tumor segmentation, section V highlights a case study with an extensive database of results obtained from public and private patient datasets. Finally, conclusions and open issues are summarized in Section VI.

II. FEDERATED LEARNING METHODS

The algorithms analyzed in this section range from vanilla FL tools, such as Federated Averaging (FA), relying on the orchestration of the PS, to fully decentralized FL, namely Consensus-driven Federated Averaging (CFA), based on distributed coordination. In server-based FL systems, the data owners (i.e., the MN clients) and the global model owner (i.e., the PS) are the two major entities. On the other hand, in fully decentralized tools the PS is replaced by a consensus over the clients, namely the local model owners. For all cases, the data are distributed among N clients rather than being kept centrally, so each data owner $i = 1, \ldots, N$, has a private dataset $\mathcal{D}_i$ of size $S_i = |\mathcal{D}_i|$.

A. VANILLA FL

The FL process generally aims to obtain an optimized global model $\mathbf{w}_G$ that minimizes a global loss function $\mathcal{L}(\cdot)$ decomposed into the sum of local losses as:

$$\mathbf{w}_G = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}(\mathbf{w}) = \underset{\mathbf{w}}{\operatorname{argmin}} \left[\frac{1}{N} \sum_{i=1}^{N} \mathcal{L}_i(\mathbf{w}) \right], \tag{1}$$

with $\mathcal{L}_i(\cdot)$ being the local loss function observed by client i. Problem (1) is solved iteratively by alternating the optimization of a *local model* at each client, i.e., using a gradient-based method, with a round of communication with the PS to obtain an updated *global model*. In particular, the FL process is characterized by three main steps: *task initialization* (executed only once at beginning), *local model optimization* and *aggregation*. The *task initialization* is implemented during the first iteration, $t = 0$: the server determines the target task (i.e., the application and the data requirements), as well as the key parameters of the global model and the training process, such as the learning rate or the number of local epochs. The server then broadcasts the initialized global model $\mathbf{w}_{G,t}$ and task to the chosen participants. In the *local model optimization* phase, at iteration $t > 0$ each participant utilizes the local data $\mathcal{D}_i$ and processing capacity to update the local model parameters $\mathbf{w}_{i,t}$ based on the global model $\mathbf{w}_{G,t}$. The aim of participant i is thus to minimize the local loss function, $\mathbf{w}_{i,t} = \operatorname{argmin}_{\mathbf{w}} \mathcal{L}_i(\mathbf{w})$. This is solved via gradient methods, such as Stochastic Gradient Descent (SGD) [40]:

$$\mathbf{w}_{i,t} \leftarrow \mathbf{w}_{i,t} - \eta \nabla \mathcal{L}_i(\mathbf{w}_{i,t}; b_i), \tag{2}$$

where η is the learning rate while $\nabla \mathcal{L}_i(\mathbf{w}_{i,t}; b_i)$ represents the gradient of the loss function with respect to the model $\mathbf{w}_{i,t}$ and it is measured on a data mini-batch $b_i \subseteq \mathcal{D}_i$. The optimized local parameters $\mathbf{w}_{i,t}$ are sent to the PS to be aggregated. In the following *aggregation* step, the PS collects the local models from the clients and feds back an updated version of global model parameters for the next iteration $t+1$, namely $\mathbf{w}_{G,t+1}$. Main aggregation policies are reviewed in [11, Chapter 3]. Finally the clients use the updated global model to update the local optimization (2). The *training rounds*, consisting of the above described local model optimization and aggregation steps, are repeated until each model $\mathbf{w}_{i,t}$ converges to $\mathbf{w}_G$, or a desired training accuracy is obtained.

B. FEDERATED AVERAGING WITH TRUSTED PS

The main parameters to control the computational effort of FL are: the percentage (C) of clients who take part in an update cycle, the number (E) of local epochs executed by each client and the mini batch b_i size (B) used for each local update. The latter one, considering the different resources that each MN may have, can be relaxed and optimized differently in each node. In what follows, the Federated Averaging (FA) algorithm introduced by [41] is tuned for the medical imaging problem.

In the *local model optimization* step, the client runs, for a number of local epochs E, the Adaptive Moment Estimation (Adam) optimizer [42], that exploits first and second order moments to overcome local minima:

$$\mathbf{w}_{i,t} \leftarrow \mathbf{w}_{i,t} - \eta \frac{\sqrt{1-\beta_2^n}}{1-\beta_1^n} \frac{\mathbf{m}_{i,n}}{\sqrt{\mathbf{v}_{i,n}} + \sigma}. \tag{3}$$

β_1 and β_2 are two hyperparameters and the decaying averages $\mathbf{m}_{i,n}$ and $\mathbf{v}_{i,n}$ are computed respectively as follows:

$$\mathbf{m}_{i,n} \leftarrow \beta_1 \mathbf{m}_{i,n-1} + (1-\beta_1)\nabla \mathcal{L}_i(\mathbf{w}_{i,t}; b_i) \tag{4}$$

$$\mathbf{v}_{i,n} \leftarrow \beta_2 \mathbf{v}_{i,n-1} + (1-\beta_2)\nabla^2 \mathcal{L}_i(\mathbf{w}_{i,t}; b_i), \tag{5}$$

where n is the timestep index of the Adam optimizer. Notice that SGD is not recommended on complex models as it needs careful tuning of the learning rate as the training progresses. For image segmentation problems, i.e., tumor segmentation, the number of local epochs E is usually kept small ($E = 1, 2$), as already verified in other works [13], while the batch size B is increased as much as possible to exploit all the parallel computations given by the Graphics Processing Units (GPUs).

The *aggregation* step at round t is performed through a weighted average according to the number of samples $S_i = |\mathcal{D}_i|$ of each client:

$$\mathbf{w}_{G,t+1} = \frac{\epsilon}{\sum_{j=1}^{N} S_j} \sum_{i=1}^{N} S_i \mathbf{w}_{i,t} + (1-\epsilon)\mathbf{w}_{G,t}, \tag{6}$$

where ϵ regulates the memory of the past models in order to have smoothed, or less rapid, changes of the weights. The *aggregation* step is a crucial part of the algorithm and affects the performances. Studies on the adaptive weight function, that regulates the importance of the client's contribution in the global model, have been performed in [19]. However, in the context of medical imaging, this method is less effective since the datasets in the MNs present few variations of the brightness and/or the noise figure.

Much attention instead should be paid to the characterization or customization of the models in each client. For example, FedPer, proposed in [43] splits the layers of the deep learning model into baseline and personalized ones. While the basic layers are collaboratively learned using the traditional FL technique, the personalized ones are learned locally and not shared, i.e. opportunistically, allowing more flexible training of multiple tasks. Adaptation of this technique to more complex models employed for brain tumor segmentation (Section III) should focus on the *encoder* part, which is generally a pre-trained classification network like VGG [44]/ResNet [45]. On the other hand, the *decoder* part could host the personalization layers [46].

C. CONSENSUS-DRIVEN FEDERATED AVERAGING (CFA)

The FA policy discussed in Section II-B relies on the PS orchestration and may be subject to privacy concerns, especially if the PS infrastructure is vulnerable (e.g., untrusted). In these scenarios, medical sites might avoid joining the collaborative training process to protect their privacy-sensitive data despite the benefits introduced by the cooperation. An alternative approach explored in the following is the consensus-driven FA strategy, namely CFA, that provides a solution to the FL problem (1) using a fully distributed and adaptive approach.

In particular, the CFA consists again of an *aggregation* and a *local model optimization* step: however, differently from the server-based FL, both steps are implemented by the MNs on each learning iteration. CFA is thus serverless as the MNs can cooperate with one another without the coordination of the PS. Rather than sending the model updates to the PS, the medical sites can directly forward the ML model parameters to neighboring participants, provided that they are authenticated as members of the pool of FL learners, and using peer-to-peer communication

Algorithm 1 Consensus-Driven FA

```
1:  procedure CFA($\mathcal{N}_{i,t}$)                          ▷ Run on client $i$
2:      authentication with network broker
3:      receive parameters $(\eta, E, B)$                        ▷ RX from broker
4:      initialize $\mathbf{w}_{i,0} \leftarrow$ device $i$
5:      initialize $\mathbf{m}_{i,0} \leftarrow 0$
6:      initialize $\mathbf{v}_{i,0} \leftarrow 0$
7:      initialize $n \leftarrow 0$                              ▷ Adam timestep
8:      for each round $t = 1, 2, \dots$ do                      ▷ Training loop
9:          receive$\{\mathbf{w}_{k,t}\}_{k \in \mathcal{N}_{i,t}}$    ▷ RX from broker
10:         $Dec\{\mathbf{w}_{k,t}\}_{k \in \mathcal{N}_{i,t}}$        ▷ Decipher weights
11:         equation (7)                                          ▷ Aggregation step
12:         $\mathbf{w}_{k,t} =$ ModelUpdate$(\boldsymbol{\psi}_{i,t})$
13:         send $Enc\left(\mathbf{w}_{i,t}\right)$              ▷ Encrypt and TX to broker
14:     end for
15: end procedure
16: procedure ModelUpdate($\boldsymbol{\psi}_{i,t}$)               ▷ Model opt. step
17:     $\mathcal{B} \leftarrow$ mini-batches of size $B$
18:     for each local epoch $j = 1, 2, \dots, E$ do
19:         for batch $b \in \mathcal{B}$ do                     ▷ Local Adam
20:             $n \leftarrow n + 1$
21:             $\mathbf{m}_{i,n} \leftarrow \beta_1 \mathbf{m}_{i,n-1} + (1-\beta_1)\nabla\mathcal{L}_{i,t}(\boldsymbol{\psi}_{i,t})$
22:             $\mathbf{v}_{i,n} \leftarrow \beta_2 \mathbf{v}_{i,n-1} + (1-\beta_2)\nabla^2\mathcal{L}_{i,t}(\boldsymbol{\psi}_{i,t})$
23:             $\boldsymbol{\psi}_{i,t} \leftarrow \boldsymbol{\psi}_{i,t} - \eta \frac{\sqrt{1-\beta_2^n}}{1-\beta_1^n} \cdot \frac{\mathbf{m}_{i,n}}{\sqrt{\mathbf{v}_{i,n}+\sigma}}$
24:         end for
25:     end for
26: end procedure
```

links. The MNs implement an ad-hoc aggregation step that incorporates into the local model adaptation the information collected from the local neighborhoods. Such aggregation is typically based on consensus [18], [47] or gossip methodologies [48], [49].

The pseudo-code of the CFA can be found in Algorithm 1. First, local model optimization (3) is performed using local data $\mathcal{D}_i$ over a number E of local rounds/epochs which can be tuned depending on the MN computing and energy requirements. The updated model is then sent to neighbors. In the aggregation step, each MN client i implements the average consensus policy to obtain the aggregated model $\boldsymbol{\psi}_{i,t}$ with the help of the neighbors:

$$\boldsymbol{\psi}_{i,t} = \mathbf{w}_{i,t} + \frac{\epsilon_t}{\sum_{j \in \mathcal{N}_{i,t}} S_j} \sum_{k \in \mathcal{N}_{i,t}} S_k \left(\mathbf{w}_{k,t} - \mathbf{w}_{i,t}\right), \quad (7)$$

where ϵ_t controls the stability of the update and $\mathcal{N}_{i,t}$ contains the neighbors of client i at round t. Notice that similarly as for server-based FL, each client might either defer the model aggregation until the neighbors complete their local model optimization (synchronous implementation) or rather apply the model aggregation as soon as they complete their model optimization, regardless of neighbors status (asynchronous implementation). These aspects are analyzed in more detail in the next sections.

D. COMPARATIVE ANALYSIS AND PRIVACY CONSIDERATIONS

Vanilla FL typically assumes that the PS and the clients are all honest, meaning that the clients are training with their own private data in a good faith, and send true local models to the PS. However, since the PS aggregates the models from all clients, it is an appealing target for possible attackers and thereby a single point of failure in the distributed platform. So, despite the fact that FL can prevent user privacy leaks, the parameter server is nevertheless subject to threats such as server-side training sample reconstructions [9]. Therefore, trusted implementations of the PS are of particular relevance in order to be suitable for the medical imaging problem. For example, differential privacy techniques add a noise term to each local model in order to prevent information from being exposed during the model exchange [50], [51]. The problem of untrusted PS is tackled in this paper at network layer using encrypted and authenticated communications on each learning iteration, in exchange for larger computation/communication overhead.

Besides vanilla FL tools, fully decentralized CFA replaces the PS with consensus and in-network processing directly between the clients (Sect. II-C). Differently from server-based FL, where the PS stores the local models of *all* the participating clients, in CFA the clients implementing consensus are owners of a (small) subset of the local models, namely the ones shared by the neighborhood $\mathcal{N}_{i,t}$. Thereby, it is unlikely that a model inversion attack targeting an individual client could reconstruct the training samples of all the learners. Furthermore, although the CFA architectural approach solves the untrusted PS issues, the untrusted client problem still needs to be carefully considered. For example, distributed ledger technologies, such as blockchain, provide an effective approach for removing the PS, that is vulnerable to attacks [28]. Moreover, it can be also exploited to address the problem of untrusted clients [52], ensuring security of local model updates obtained from authenticated clients. The development of robust decentralized FL designs against adversarial manipulations or data poisoning is however an open problem [53].

III. BRAIN TUMOR SEGMENTATION: FL MODELS AND METRICS

This section describes the brain tumor segmentation and classification task. In particular, we highlight the ML model selected for FL processing as well as relevant loss and accuracy metrics. Tumor segmentation is one of the fundamental tasks in medical diagnosis to support radiologists and clinicians, and also to reduce idle times for evaluating potential treatments. Given a set of MRI slices, the goal of brain tumor segmentation is to extract regions of interest that capture the tumor extent and its shape. This is accomplished by assigning a class label for every pixel (or voxel) in the images, indicating the presence/absence and/or the morphology of the tumor. As an example, Fig. 2 reports

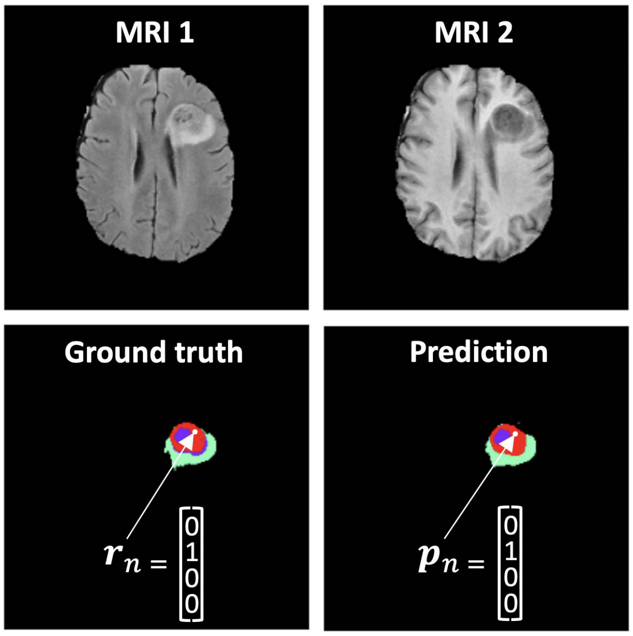

MRI 1
MRI 2
Ground truth
Prediction
$r_n = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$
$p_n = \begin{bmatrix}0\\1\\0\\0\end{bmatrix}$

FIGURE 2. Brain tumor segmentation with 2 input layers and 4 segmentation levels: MRI typology 1 (up-left), MRI typology 2 (up-right), ground-truth $r_n = [r_{\ell,n}]_{l=1}^{L}$ (down-left), prediction $p_n = [p_{\ell,n}]_{l=1}^{L}$ (down-right).

the two input MRI modalities on the top, while the expected ground-truth and predicted tumor segmentation labels are in the bottom-left and bottom-right, respectively. Achieving accurate segmentation labels in this context requires the definition of complex processing systems capable of handling MRI scans with different spatial resolutions, modalities, and levels of noise, according to the specific medical equipment employed.

A. U-NET MODEL AND FL ADAPTATIONS

Current state-of-the-art ML systems for brain tumor segmentation heavily rely on Convolutional Neural Networks (CNN) architectures. Most notably, the U-Net model [25] has been gaining popularity in recent years thanks to its outstanding performances, especially for medical segmentation tasks. The U-Net architecture is composed by two parts: the *encoder*, which extracts the context from the images, and the *decoder*, whose task is to determine the segmentation region. The encoder uses several convolutional blocks, followed by max-pooling operations, for encoding the input image into intermediate representations at different spatial resolutions. On the other hand, the decoder is a symmetric network, composed by the same structural form of the encoder, that performs upsampling and concatenation operations for extracting the final segmentation from the input image.

In this paper, we employ the U-Net model [27] that modifies the original one [25] and it is better suited to the considered FL medical context. Compared to [25], the encoder introduces dropout layers for allowing better model generalization and prevent overfitting. Moreover, the model has been adapted to receive input images with different number of channels, depending on the available MRI modalities, while also providing up to four segmentation levels. In total, it presents about 7.8 millions of parameters and weights of size 30 MB.

B. DICE LOSS AND SIMILARITY METRICS

The ML model is trained using a combination of two losses, namely the Generalized Dice Loss (GDL) [54] and the Cross Entropy (CE). The GDL $\mathcal{L}_{GDL}$ is a generalization of the conventional Dice Loss that takes into account multiple class segmentation problems, rather than binary ones. Considering L segmentation labels and K image elements, the GDL can be computed as:

$$\mathcal{L}_{GDL} = 1 - 2\frac{\sum_{\ell=1}^{L} w_D^{(\ell)} \sum_{n=1}^{K} p_{\ell,n} r_{\ell,n}}{\sum_{\ell=1}^{L} w_D^{(\ell)} \sum_{n=1}^{K} p_{\ell,n} + r_{\ell,n}}, \tag{8}$$

where $r_{\ell,n} \in \{0, 1\}$ and $p_{\ell,n} \in \{0, 1\}$ are respectively the voxel values of the reference foreground segmentation (ground truth) and the values of the predicted map for the foreground label $\ell = 1, .., L$ (an example is given in Figure 2 with $L = 4$). $w_D^{(\ell)}$ is the weight to model the contribution of each label, and it is defined as $w_D^{(\ell)} = \frac{1}{(\sum_{n=1}^{K} r_{\ell,n})^2}$. Finally, the CE loss $\mathcal{L}_{CE}$ is:

$$\mathcal{L}_{CE} = \sum_{n=1}^{K}\sum_{\ell=1}^{L} r_{\ell,n} \log(p_{\ell,n}). \tag{9}$$

Note that the formulations of (8), (9) consider the ground-truth $r_{\ell,n}$ and the label $p_{\ell,n}$ segmentations encoded as one-hot representations. Finally, the total loss in (1) can be computed as:

$$\mathcal{L} = \lambda\mathcal{L}_{GDL} + (1-\lambda)\mathcal{L}_{CE}, \tag{10}$$

with $\lambda = 0.85$ and is used to update the weights of the Neural Network (NN).

For assessing the quality of the trained models for brain tumor segmentation, we use the Dice Similarity Coefficient (DSC) metric [55]. Given a pair of ground-truth $\mathcal{T}$ and predicted segmentation patches $\mathcal{P}$, the DSC ranges from 0 to 1 and it is computed as:

$$\text{DSC} = \frac{2|\mathcal{P} \cap \mathcal{T}| + 1}{|\mathcal{P}| + |\mathcal{T}| + 1} \tag{11}$$

where $|\mathcal{P} \cap \mathcal{T}|$ denotes the intersection between the predicted and ground-truth patches, while $|\mathcal{P}|$ and $|\mathcal{T}|$ are the cardinalities of the predicted and true segmentation, respectively. High quality models should possess a DSC value close to 1, indicating a near-optimal match between ground-truths and predictions. Of course, since the DSC works with binary masks, a coefficient will be generated for each of the segmented parts of the tumor (e.g., tumor-core and not-tumor-core).

IV. NETWORK ARCHITECTURE AND MQTT PROTOCOL

In this section, a network architecture is proposed to integrate the FL tools described previously. The proposed system adopts the MQTT transport protocol to coordinate the real-time exchange of the U-NET model parameters through MQTT-compliant *publish* and *subscribe* operations. We discuss the benefits of the chosen transport layer, as compared with HTTP based representational state transfer (REST) services, and design an optimized set of information to be embedded into the MQTT payload, as well as model parameters compression, serialization and Quality-of-Service (QoS) mechanisms. The proposed architecture is validated in Sect. V targeting the brain tumor segmentation task (Sect. III). Nevertheless, the proposed framework is general enough for application to distributed training of deep neural network models.

The aggregation and local model optimization steps of the FL process can be implemented via *synchronous* or *asynchronous* policies: when the training process adopts the synchronous orchestration, all the network entities (clients and PS, if any) share the same time reference t: therefore, the model aggregation should wait for all the scheduled clients to complete their local model optimization. Notice that the aggregation steps on the PS (6) and on the client, in fully decentralized training (7), can be adopted as they are. On the contrary, in the asynchronous orchestration, the clients, or the PS, aggregate the available local models without any regard of their relative temporal alignments. More specifically, asynchronous model aggregation (6) implemented on the PS at time t becomes

$$\mathbf{w}_{G,t} = \frac{\epsilon}{\sum_{j=1}^{N} S_j} \sum_{i=1}^{N} S_i \mathbf{w}_{i,t-\tau_i} + (1-\epsilon)\mathbf{w}_{G,t-T_{PS}}, \quad (12)$$

with $\mathbf{w}_{i,t-\tau_i}$ being the local model from client i available at time $t - \tau_i$, $\tau_i \neq 0$ the relative timing mismatch with the PS, and T_{PS} regulating the time span between two global model updates. Similarly, in the asynchronous implementation of the CFA algorithm, once a medical center finishes its local model optimization, it immediately switches to the aggregation step, regardless of whether its neighbors have already finished their local model optimization or not. Therefore, each MN i will now receive from neighbors k the last uploaded models of rounds $t - \tau_k$, $\forall k \in \mathcal{N}_{i,t}$, namely, from (13)

$$\boldsymbol{\psi}_{i,t} = \mathbf{w}_{i,t} + \frac{\epsilon_t}{\sum_{j \in \mathcal{N}_{i,t}} S_j} \sum_{k \in \mathcal{N}_{i,t}} S_k \left(\mathbf{w}_{k,t-\tau_k} - \mathbf{w}_{i,t}\right). \quad (13)$$

In what follows, and based on the above considerations, we analyze in detail four different network architectures and related MQTT orchestration mechanisms weaving together synchronous and asynchronous operations, i.e., on the clients and/or the PS.

A. MQTT MESSAGING AND FL PROCESS ORCHESTRATION

The choice of the MQTT protocol [56], over for example the HTTP RESTful, was dictated by many factors, starting with the superior bandwidth efficiency of MQTT and lower latency [57]. Another important aspect is the low overhead of the protocol, designed for low-power machine-to-machine transactions, which is crucial given the size of the messages exchanged during the federated training phase. The MQTT protocol is also suited for one-to-many communications as needed i.e., during the distribution of the global model from the PS to the MN. Finally, the architecture can be easily extended to the Internet-of-Things (IoT) field, i.e., on embedded devices, or smartphones, where the implementation is practically constrained by low computational capabilities and battery usage requirements.

1) MQTT MESSAGING

In the proposed FL architecture, both the PS and the medical nodes/centers members of the federation act as MQTT client devices and support publish and subscribe operations. Devices thus share the layers of the deep learning model by encapsulating the parameters into the MQTT standard payload. In particular, the basic subset of information included in the payload are:

i) the updated weights $\mathbf{w}_{i,t}$ of the U-NET model trainable layers (Sect. III.A),

ii) tunable parameters for monitoring the convergence, namely: the number E of local rounds to be executed in each client, the learning rate η to be used in the local model optimization step, the target DSC performance (Sect. III.B) and the patience (to apply the early stopping procedure),

iii) training statistics: the client identification number, the federated round indicator and the performance metrics, i.e., DSC in (11), obtained from the validation dataset.

2) PUBLISH-SUBSCRIBE OPERATIONS AND QoS

MQTT publishing and subscription operations are organized into a number of topics. Considering both FA and CFA algorithms, the main topics are the ones related to the MN and PS exchange of the model parameters. In case the model size exceeds the maximum dimension of the MQTT messages (by default set to 250 MByte), the model can be automatically decomposed in different fragments, each representing one or multiple layers of the model, and published separately. Other topics are related to the configuration parameters, to the number of samples of each medical center and to a timestamp that indicates the last time τ_k the k medical center has been seen. Before being sent to the broker, all messages are first serialized into binary objects with the *cPickle* module. We chose to use this module rather than the classic JSON serialization method because of its higher speed and flexibility [58]. Messages are further compressed using the *zlib* module, in order to occupy less space and bandwidth (about 10% less), and are sent to the MQTT broker with Quality of Service (QoS) 2. In particular, QoS 2 implements a 4-way handshake mechanism that is especially effective over communication links with poor quality, while adding a negligible delay (100 ms) if compared with the time required for transferring the message payload (>1 s). Minimizing packet losses is critical in FL while, as also shown in [59], QoS 2 is the most effective choice for large payload MQTT publishing operations. For all cases, the TLS protocol is adopted to encrypt the exchanged messages.

3) MQTT BROKER

Considering FA, the clients publish their data to a *MQTT broker* that is in charge of maintaining the model parameters and forwarding them to the PS whenever a change is detected. Multiple copies of the messages, as well as packet losses, are handled directly by the QoS 2 specification. The MQTT broker service thus acts as sink node for local models collection and it is thus maintained until the end of the training process. According to the type of broker, the memory size and capabilities can change: in our case, we used a single-threaded broker but other options are also possible. On each round, the broker accepts subscriptions from the active clients that publish their model parameters. Considering server-based FA, all clients also subscribe to the same broker service, i.e., to download the updated global model. On the other hand, for CFA, each client subscribes to its neighbors' topics to retrieve their last available models and publishes the updated model on its related weights topic.

The software that implements the FL process and enables the MQTT transactions for the client (and the PS) is available online [60]. Once installed, it is completely self-sustaining while all network entities maintain an idle mode state until the training process is initiated or updated. The FL process begins with a Command&Control (C&C) tool that uploads the main parameters on the MQTT broker and starts the training. Once the process is started, the PS or the clients, in case of fully decentralized implementation, are in charge of maintaining the training. Furthermore, new authenticated MNs that aim to join the federation, are accepted from any geographical location. The number of resources consumed are chosen a priori according to the complexity of the task and the dataset, the MN computing capabilities and to the desired training speed. To stop the process many possibilities are accepted: reaching a target number of federated rounds or performance, or a direct message from the C&C tool.

B. SYNCHRONOUS AND ASYNCHRONOUS FL NETWORK ARCHITECTURES

The development of a network architecture designed for native FL support over the Internet needs to face two critical challenges. First, slower clients, i.e., retaining large datasets (high number of MRI images) or characterized by low computational resources, experience a longer training time and might penalize faster clients (straggler effect). Second, global model updates issued by the PS, if used, or by the clients, when implementing consensus, should avoid possible deadlock situations caused by packet losses, i.e., links with poor quality as well as delayed model updates from neighboring clients.

To overcome these issues, we introduce and analyze different network architectures that leverage distinct levels of a/synchronicity on the client or the PS (if used), respectively. We show that introducing temporal variations and asynchronicity over some of the FL network entities, namely the clients and the PS, can lead to more efficient training in the presence of slow/heterogeneous FL learners. In particular, for the proposed implementation, a client is considered asynchronous if it can perform more than two local rounds without stopping or waiting for global/local model updates from the PS or neighboring clients. The time span of one local round implemented on client k is defined as $T_{round}(k)$, which is the sum of the time required to download the weights from the MQTT broker, training the new model (local model optimization step), encrypt, compress and upload the weights to the broker:

$$T_{round}(k) = T_{download} + T_{training} + T_{upload}. \tag{14}$$

Considering PS based FL, we implemented a timer that fires every server sleep time, namely T_{PS}. As shown in (12), when this happens, the PS decides whether to update the global model or not, depending on the backlog of local models retained by the MQTT broker. In particular, the Retain Flag of the MQTT protocol is set to true so that the last message sent by the MN is stored into the MQTT broker: when the PS (or another client) subscribes to the same topic, the broker delivers the message. In what follows we highlight 4 selected architectures: notice that three of them are based on the PS a/synchronous orchestration, while the last one supports the fully decentralized FL tools and the consensus process.

1) PS SYNCH., CLIENT SYNCH. (PS-S/C-S)

The architecture, represented in Figure 3b corresponds to the vanilla FL implementation proposed in [4], [8]: both the PS and the clients are synchronous (S) with respect to the training process, therefore all network elements share a common sense of time, while the FL process is supervised by the MQTT broker. The PS waits until all the clients complete their model optimization steps, and monitors the PS weights topic on the MQTT broker. Once all clients have published their weights, the PS is unlocked and implements the aggregation step as in (6). After the PS has published the updated global model, the clients are unlocked and the PS returns to wait.

2) PS SYNCH., CLIENT ASYNCH. (PS-S/C-A)

As described in Figure 3c, the PS keeps the same synchronous behavior as in the PS-S/C-S architecture. On the other hand, the clients adopt asynchronous (A) actions: when their model optimization step is completed, they might continue the training using local data unless the PS global model is updated. In such case, they stop the local model optimization step and replace the local weights $\mathbf{w}_{k,t}$ with the updated global model $\mathbf{w}_{G,t}$.

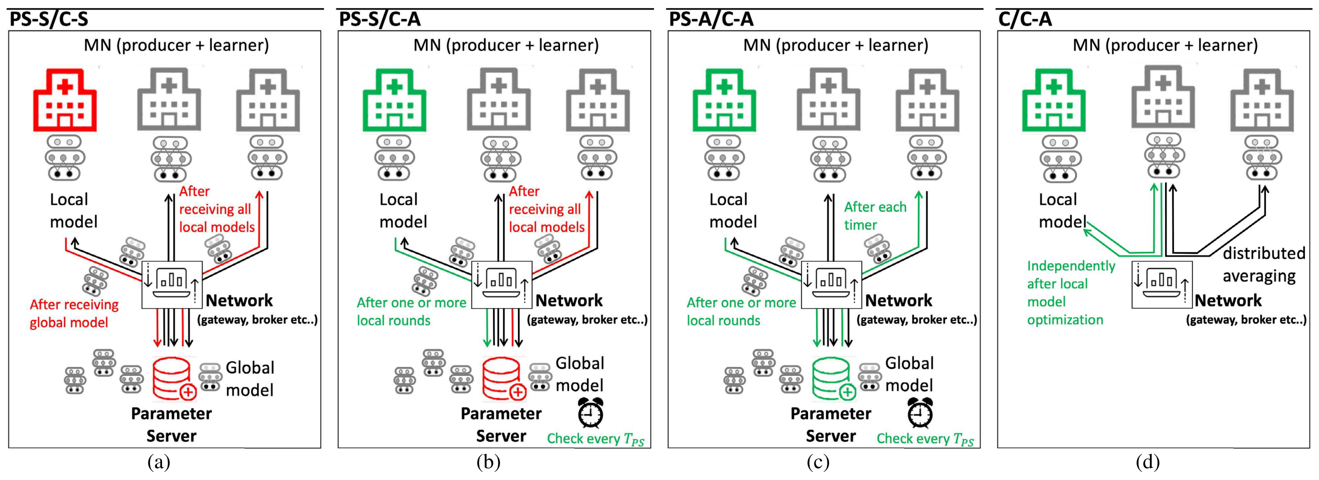


FIGURE 3. FL architectures: from left to right PS-S/C-S, PS-S/C-A, PS-A/C-A and C/C-A. Red color indicates a synchronous element (client or PS) and its implications. On the contrary the asynchronous element is depicted with green color.

3) PS ASYNCH., CLIENT ASYNCH. (PS-A/C-A)

The last type of PS-based architecture is pictured in Figure 3d. In this case, both the PS and the clients are asynchronous (A). On every T_{PS} sec., the PS collects the local model weights from the MQTT broker and updates the global model even though not all clients have finished their model optimization steps. The clients are also asynchronous and act similarly as in the PS-S/C-A architecture.

4) CONSENSUS-DRIVEN (NO PS), CLIENT ASYNCH. (C/C-A)

The architecture supports the consensus-based (C) FL and coordinate the local model exchange among the clients. As illustrated in Figure 3d, the implemented architecture is fully decentralized while every client is asynchronous. The clients can communicate directly with each other uploading and downloading the updated local model weights from the MQTT broker through the specific topics of the neighbors. In particular, the MQTT broker acts as a bridge allowing the communication among interconnected MN, i.e., possibly located in different countries, and according to an assigned connectivity matrix. When a client has finished its round, it downloads the weights of its neighbors, updates the local model according to a specified algorithm (i.e., the CFA described in Sect. II), and in turn publishes the updated local model on its weights topic.

V. CASE STUDY: DIAGNOSTIC IMAGING FOR BRAIN TUMOR SEGMENTATION

The study and validation of the proposed architecture and FL tools is performed on a federation of 5 MNs distributed across Europe. As detailed in Sect. III, we selected the task of binary segmentation of brain tumors (and tumor-like pathologies) in the axial slice-based single-channel FLAIR MR images to concentrate our effort on the challenge of implementing a networking and computing environment in a realistic clinical setting. The FL process and model quality are verified first using the BraTS public data repository (BraTS 2018), next we analyze a more realistic set up featuring an additional real-world clinical data set of images not prepared ad-hoc for testing. Such new dataset is referred to as "Athens set" and it is used to complement the federation with additional MNs. Finally, validation on the BraTS 2020 set is also considered. For all setups, the model quality, here also referred to as performance level, is measured in terms of DSC metric defined in (11).

The approach we follow for the analysis is threefold. First, the data pre-processing pipeline (Sect. V-A) targets the harmonization of BraTS and Athens sets. All the proposed FL network orchestration mechanisms, namely the PS-S/C-S, PS-S/C-A, PS-A/C-A, C/C-A, are then compared with the benchmark centralized ML (Sect. V-B) with respect to the DSC metric. In particular, the parameter server sleep time T_{PS} is optimized for asynchronous FL, targeting the PS-A/C-A architecture. Finally, we analyze the performance in a practical scenario where FL is implemented on heterogeneous medical nodes located in Italy and Switzerland (Sect. V-C). For these last experiments, we consider different combinations of training and validation sets quantifying for each case the benefits of federation.

A. DATASETS AND DATA PREPARATION

To verify the performance of the proposed FL tools and architectures we first populated 4 MNs, located in different countries, with shards from publicly available BraTS 2018 and BraTS 2020 datasets [21]–[23]. The division of BraTS dataset into training and validation sets was performed on a per-examination basis, so all slices from one examination ended up in the same set. The number of examinations and slices per node can be seen in Table 1. In particular, BraTS 2018 dataset was splitted into three shards (namely MNs 1, 2, and 3), and the new examinations from BraTS 2020 (added on top of BraTS 2018), were used for the last client (MN 4).

TABLE 1. Distribution of BraTS datasets across three different FL clients.

	Exam.	Abnorm. slices	Normal slices
Training - MN 1 (BraTS 18)	67	4421	5964
Training - MN 2 (BraTS 18)	67	4458	5927
Training - MN 3 (BraTS 18)	66	4296	5934
Training - MN 4 (BraTS 20)	59	3817	5328
Validation - MN 1 (BraTS 18)	19	1309	1636
Validation - MN 2 (BraTS 18)	19	1272	1673
Validation - MN 3 (BraTS 18)	19	1207	1738
Validation - MN 4 (BraTS 20)	25	1681	2194
Total	341	22461	30394

TABLE 2. Size and distribution of Athens' dataset into training, validation and testing sets.

	Abnormal		Normal	
	Exam.	Slices	Exam.	Slices
Training - MN 5 (Athens)	21	1173	18	1005
Validation - MN 5 (Athens)	5	233	4	186
Total	26	1406	22	1191

As previously mentioned, to complement the federation with additional nodes and training data, we considered a new private dataset (Athens dataset) consisting of 48 MRI examinations, out of which 26 contain abnormalities, and the rest 22 are normal, serving as negative control in the experiments. Similarly as for BraTS, the examinations were divided into training and validation sets on the per-examination basis. The setup has been designed purposely to cope with harmonized, but significantly different, data across nodes. The new dataset is hosted in the MN labelled as 5 while the total size of the dataset and the portions belonging to training and validation sets are depicted in Table 2. Hardware specifics and computing capabilities of the 5 deployed clients are further detailed in Table 3.

1) BRATS VS. ATHENS SET

Besides the fact that images in Athens and BraTS datasets come from different sources, they differ also in few other critical aspects. First, Athens data consist of raw samples as they come out of the MRI machine, without any skull-stripping or transformations. On the other hand, BraTS images are already pre-processed: they are all registered to the common atlas, interpolated and subsampled to an uniform resolution 1 mm^3 in all three main axes in 3D. Furthermore, BraTS collection contains only examinations with abnormalities, while in the Athens dataset there are also normal examinations (healthy patients) and a wider range of oncological cases w.r.t. the shape, position and typology. Finally, BraTS dataset contains only high- and low grade gliomas (HGG and LGG), while Athens examples feature also metastatic and smaller tumor-like lesions (see the examples in Fig. 4).

2) DATA HARMONIZATION AND AUGMENTATION

To harmonize the BraTS and Athens inputs, so that the data of all nodes can contribute to model training, we employed few normalization steps in the data processing pipeline.

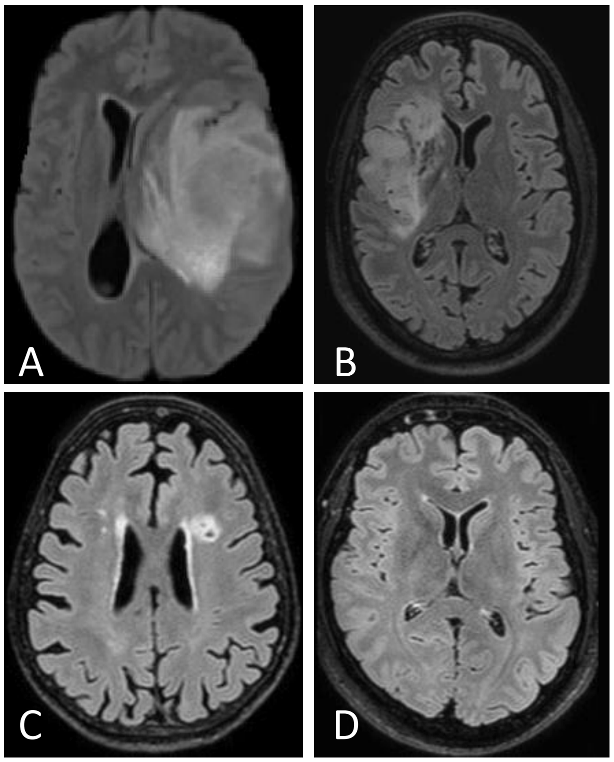


FIGURE 4. Representative images from BraTS (A) and Athens' (B, C, D) datasets: A) large detection with surrounding edema in the left hemisphere from BraTS 2020; B) similar case in right symmetrical position from Athens' data set; C) small detection on the left semi-oval centrum, and D) normal examination.

First, Athens images are resampled to the spatial resolution of 1 mm^2, so that the spatial dimensions are compatible with BraTS samples. Then, the intensities of each slice are standardized (i.e., transformed such that the mean intensity is 0 with standard deviation 1). All slices are then clipped or padded to obtain images with uniform dimensions 240×240 pixels. Because the BraTS dataset contains more fine-grained segmentation labels (marking individual parts of the tumors) compared to what we needed, we merged all labels together to produce a *whole-tumor* segmentation mask. On top of the harmonization of the input data, we employed also a data *randomization* step that consists of randomized transformations like image flipping, rotations and elastic deformations. Input images are also altered by adding Gaussian noise. This process helps to de-correlate the training samples and in turn improve the robustness of the model against overfitting. It is worth to mention that higher spatial resolution MRIs provide critical anatomical features that help to better detect illness and make diagnoses. Unfortunately, High Resolution (HR) MRIs are hampered by extended scan times and low signal-to-noise ratio (SNR), especially when hardware capacity is restricted. Consequently, often Low Resolution (LR) images are taken. Recent research has shown that using CNNs and single image super-resolution (SISR) techniques, HR images may be reconstructed from LR ones [61].

TABLE 3. Medical Nodes hardware and computing capabilities.

	CPU	RAM	GPU
Milan site 1: Desktop PC	Intel(R) Core(TM) i7-3930K @ 3.2 GHz	32 GB	NVIDIA GeForce RTX 3090 with 24 GB
Milan site 2: Laptop PC	i7-10750H @ 2.6 GHz	32 GB	NVIDIA GeForce GTX 1650 Ti with 4 GB
Geneva site 1: Desktop PC1	Intel(R) Core(TM) i7-6700 @ 3.4 GHz	32 GB	NVIDIA Quadro K2200 with 4 GB
Geneva site 2: Desktop PC2	Intel(R) Xeon(R) E5-1630 v3 @ 3.70 GHz	32 GB	NVIDIA GeForce RTX 3090 with 24 GB
Geneva site 3: Desktop PC3	Intel(R) Xeon(R) E5-1630 v3 @ 3.70 GHz	32 GB	NVIDIA GeForce RTX 3090 with 24 GB

B. ASSESSMENT OF NETWORK ARCHITECTURES

In the following initial tests, we deployed 4 clients co-located with the Milan site 1 (hardware specifics are detailed in Table 3). Three clients use training data from the BraTS 2018 while the remaining one uses the Athens dataset. In addition, the FL model quality is validated with both Athens and BraTS validation sets and assessed using the DSC metric (11). This is obtained by averaging the DSC metric over the full validation dataset in each MN. For real-time evaluation of the FL process, the hardware platform hosting each client has been adapted to use 4 GB of GPU memory and about 3.5/4 GB of RAM. To optimize memory usage, we adopted the TFRecord binary format and only a suited number of training samples were kept in memory: about 1000 training slices for shuffling reasons. Regarding the training parameters, we set the number E of local epochs to 1, for the reasons described in Section II-B, and the batch size B to 16. The learning rate of the Adam optimizer was set to 10^{-4}, while the β_1 and β_2 hyperparameters are fixed respectively to 0.9 and 0.999. The validation phase is performed by each MN on its validation set after receiving the global model by the PS. This validation step provides for a more accurate tracking of model quality improvements over time.

In Figure 5 we compare the DSC of PS-S/C-A and the PS-A/C-A architectures at different clock times and by varying the server sleep time T_{PS}. The optimal choice of the sleep time depends on the minimum/maximum local round time between each client, defined as:

$$\begin{aligned} T_{MAX} &= \max_k T_{round}(k) \\ T_{MIN} &= \min_k T_{round}(k). \end{aligned} \tag{15}$$

As highlighted in the corresponding scenarios, setting $T_{PS} < T_{MIN}$ improves the training time of the PS-A/C-A architecture: in particular, it reaches a target DSC of 0.85 while saving the 20% of the training time compared with the synchronous PS-S/C-A option. On the contrary, performance drops are observed when $T_{PS} >= T_{MIN}$, specifically regarding the final DSC that worsens from 0.878 to 0.865. For what concerns the PS-S/C-A architecture, setting $T_{PS} <= 2\ T_{MAX}$ (blue area), gives the worst performance. This result can be due to the fact that in the PS-S/C-A architecture, the clients perform too many local rounds before the aggregation step and this can lead to a bias in the training process. Finally, for the same reason, we can observe that increasing too much T_{PS} (red line) is detrimental and not useful.

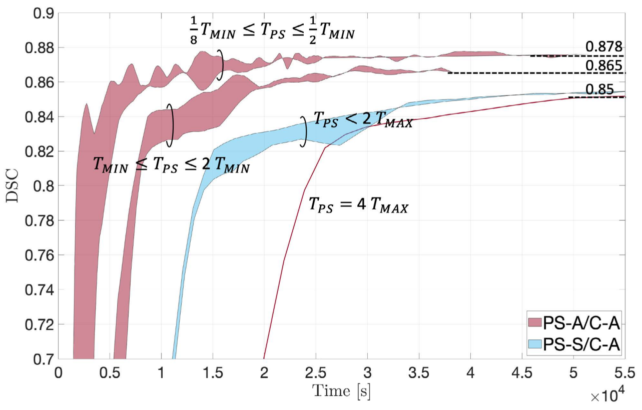


FIGURE 5. Comparison between PS-S/C-A (blue areas) and PS-A/C-A (red areas) architectures for varying T_{PS}. The plots report the observed DSC at different clock times. Highlighted scores of 0.878, 0.865 and 0.85 produce different prediction images (as analyzed in Figure 6).

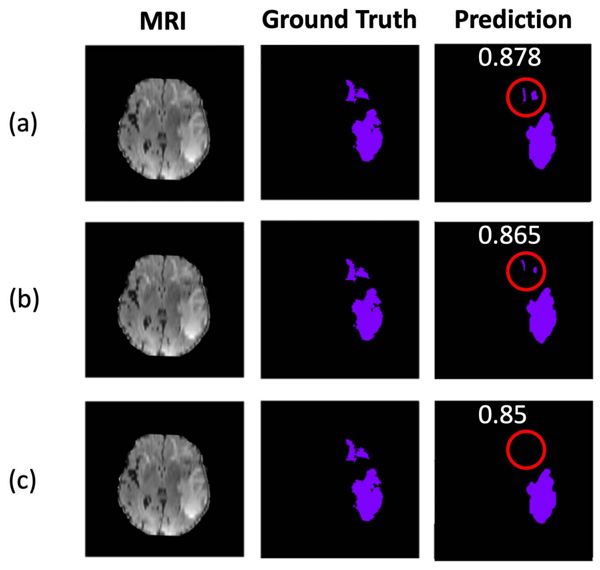


FIGURE 6. Qualitative representation of performance at convergence in Figure 5. From top to bottom, the predicted images reached a DSC metric of respectively 0.878, 0.865 and 0.85. These values correspond to prediction by, respectively, PS-A/C-A with $T_{PS} < T_{MIN}$ (image a), PS-A/C-A with $T_{PS} >= T_{MIN}$ (image b) and PS-S/C-A (image c). Red circle highlights the main detection difference in the upper part of the tumor.

Besides performances as measured by DSC and training time, in Figure 6 we analyze the tumor segmentation quality by visual inspection and considering the final DSC reached by the FL architectures PS-S/C-A and PS-A/C-A after $4 \cdot 10^4$ seconds of training. Although the DSC metric is above 0.85 for all cases, the trained models in Figure 6a

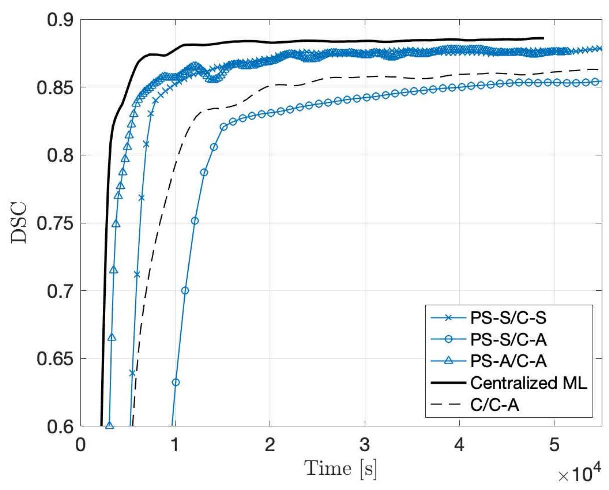


FIGURE 7. **Comparison among all architectures. The centralized ML benchmark is highlighted with a thick solid black line, while the fully decentralized FL architecture (C/C-A) is in dashed line. The three server-based FL architectures are in blue lines with different markers.**

and 6b are able to detect the upper part of the tumor as they achieve a DSC of 0.878 and 0.865, respectively. On the other hand, the model in Figure 6c cannot, since the corresponding DSC is only 0.85. This example is critical to understand how even a small increase on DSC can significantly improve the predicted image, as well as the segmentation precision/quality.

Considering the same settings described previously, in Figure 7 we investigated the differences among all the proposed FL architectures when compared with the benchmark centralized ML case. The C/C-A architecture replaces the PS in exchange for slower convergence time compared with server-based FL policies (i.e., PS-A/C-A). However, it reaches comparable dice whole metric above 0.85. The performance of the PS-S/C-A scheme is worse than the fully decentralized C/C-A: the heterogeneity of the datasets suggests the use of an asynchronous PS with $T_{PS} < 2\,T_{MAX}$. Compared with synchronous FL, the asynchronous PS-A/C-A architecture, with $T_{PS} < T_{MIN}$, obtains the best tradeoff between performances and training time.

C. IMPACT OF HETEROGENEOUS NODES AND VARYING DATA SETS

In this section we target practical setups where the MNs are located in different countries. Nodes are also equipped with varying computing capabilities, according to Table 3, and communicating over the Internet. The goal is to highlight the robustness of the FL process in handling both data and client heterogeneity. The proposed setups also verify how trained models via FL can be updated and transferred to newcomer MNs bringing new data to the process. In what follows, we adopted the PS-A/C-A architecture by keeping the same server sleep time $T_{PS} = 127$ seconds, as optimized in the previous section. The local round time of the clients varies depending on the physical machine, namely the hardware and

TABLE 4. **Results of the real-time remote experiments with the PS-A/C-A FL architecture and $T_{PS} = 127$ s. MN are described in Tables 1 and 2, sites (s.) in Milan (M.) and Geneva (G.) and corresponding hardware are in Table 3. For each of the three experiments (Tables a-b-c), we describe in the upper sub-table the different compositions of the MNs according to their physical sites and the round time T_{round}. The lower sub-table reports T_{MAX} and T_{MIN} values along with the time required to achieve a DSC of 0.85 and the DSC reached after $5 \cdot 10^4$ s.**

(a)

Test 1	MN1	MN2	MN3	MN5
Hardware	M. s. 1	M. s. 1	M. s. 1	G. s. 1
T_{round}[s]	441	441	441	332

T_{MAX}[s]	441
T_{MIN}[s]	332
Time [s] to DSC 0.85	12000
DSC after $5 \cdot 10^4$ s	0.878-0.880

(b)

Test 2	MN1	MN2	MN3	MN5
Hardware	M. s. 1	M. s. 1	M. s. 2	G. s. 1
T_{round}[s]	127	127	747	332

T_{MAX}[s]	747
T_{MIN}[s]	127
Time [s] to DSC 0.85	7000
DSC after $5 \cdot 10^4$ s	0.866-0.870

(c)

Test 3	MN1	MN2	MN3	MN5
Hardware	M. s. 1	G. s. 2	G. s. 3	G. s. 1
T_{round}[s]	80	72	70	359

T_{MAX}[s]	359
T_{MIN}[s]	70
Time [s] to DSC 0.85	6000
DSC after $5 \cdot 10^4$ s	0.8707-0.8715

the computing capability. We also used the same number of clients and dataset distributions.

As described in Tables 4a-4b-4c, we performed 3 tests using MNs 1, 2, 3 and 5 in different sites. A visual representation of test number 3 (Table 4c) indicating the specific MNs locations and corresponding dataset distributions can be found in Figure 8. Each MN is characterized by a different time round T_{round}, while the corresponding T_{MAX} and T_{MIN} (15) values are reported for simplicity. For each test, we also reported the time required to achieve a target DSC of 0.85, and the DSC reached after a training period of $5 \cdot 10^4$ seconds (corresponding to about 800 rounds). Considering the time required to reach a DSC of 0.85, we can notice that having a low T_{MIN} leads to higher performances especially in the first part of the training process. This is due to the fact that the MNs with large computing capability can perform several local rounds between two global model updates, thus refining the local model. On the other hand, the higher DSC reached after a period of $5 \cdot 10^4$ seconds is achieved only by test 1 (Table 4a) that has the optimal server sleep time: $T_{PS} < T_{MIN}$.

In the last experiment, we analyzed a further setup where the MN labelled as 4 in Table 1 joins the federation. In particular, the new node contains examples drawn from the BraTS 2020 set and is located in the Milan site 1 (Table 3). The purpose of these tests is to verify the robustness

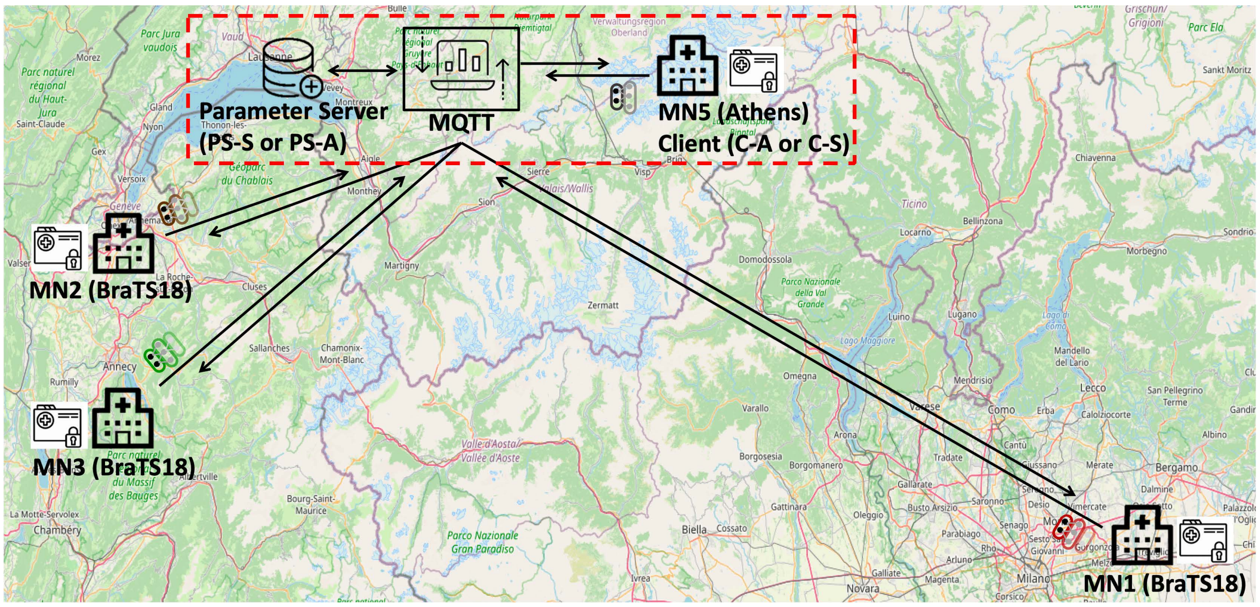


FIGURE 8. Geographical representation of Test 3 of Table 4. The MNs (i.e., the datasets) are also described in Tables 1.

TABLE 5. FL performances with 12 different training/validation set combinations ranging from BraTS 2018, BraTS 2020 and the new Athens dataset. Cross-markers indicate, for each case, the training sets and the corresponding validation examples on which the DSC (second-last column) is computed. The dimension of the total MRI training datasets is reported in the last column in terms of number of slices (of size 115 KB). Max and min values of validation DSC for each dataset are highlighted in green and red, respectively. Notice that the MN 1-2-3 (Table 1) are grouped together in BraTS 18 for brevity.

		Training			Validation					
	# MNs	BraTS 18	BraTS 20	Athens	BraTS 18	BraTS 20	Athens	DSC	# MRI slices	
1	1			X			X	0.72-0.76	2178	one dataset
2	1	X			X			0.860-0.870	31000	
3	1		X			X		0.874-0.878	9145	
4	4	X		X			X	0.730-0.750	33178	two datasets
5	4	X		X	X			0.872-0.876	33178	
6	4	X	X		X			0.877-0.879	40145	
7	4	X	X			X		0.881-0.883	40145	
8	2		X	X		X		0.878-0.882	11323	
9	2		X	X			X	0.740-0.760	11323	
10	5	X	X	X	X			0.879-0.881	42323	three datasets
11	5	X	X	X		X		0.888-0.898	42323	
12	5	X	X	X			X	0.730-0.750	42323	

of FL, in terms of DSC, for varying combinations of training and validation sets chosen from the BraTS 2018, the BraTS 2020 and the Athens data previously described. In Table 5, we compared the observed DSC after 800 FL rounds (corresponding to approx. $5 \cdot 10^4$ seconds) considering all possible combinations of training and validation sets, as indicated by the corresponding markers. In particular, the first three cases (rows 1, 2 and 3) correspond to single node training where the MN uses only its local data for learning (namely BraTS 18, BraTS 20 or Athens), without joining the federation. The following six cases (rows from 4 to 9) highlight the training performance observed when 2 to 4 MNs share their model parameters in the federation. Finally, the last three cases (rows 10, 11 and 12) show the performance of a federation of 5 MNs using all the available datasets (42323 MRI slices) and different validation examples. We can notice that in general, if a MN joins a federation (of 2 to 5 MNs), the DSC increases as expected. Although improvements are numerically small, they provide a significant increase of the predicted image segmentation resolution, as closer to the ground-truth (see also the examples of Figure 6). With respect to single node training, the DSC increases on average by 1% considering a federation of 4 MN, and scales up to 1.6% when all the training sets are considered (5 MNs). Notice that further improvements are expected by replacing the current U-NET model with more complex options [24], [26], as well as increasing the number of deployed MNs, in exchange for larger training time. Looking now at the training datasets, we can observe that the major performance improvements occur when the MNs retaining BraTS data join a federation with other MNs holding training samples with similar characteristics (BraTS 2018 or 2020): see cases 2, 6 and 3, 7. Interestingly, a MN that joins the federation and brings samples from the Athens set provides further improvements, estimated as 0.6% on average, see cases 6, 10, and 7, 11. This last observation is noteworthy because even with all the differences described in Section V-A and with the fewer examples/slices of Athens dataset, the increase of DSC is significant. FL performance improvements with increasing dataset size, i.e., from one to three datasets as in cases 2, 10 and 3, 11, can reach up to 1.7%, while the benefits of joining the federation are evident. From a different perspective, excluding MNs from the federation, even though they exhibit straggler-like behaviors, is not recommended since the global model might lose its ability to generalize, i.e., being less adaptive to new data. Finally, it is worth noticing that the DSC on the Athens validation dataset seems to be little influenced by the MNs holding the BraTS sets (cases 1, 4, 9 and 12). A cause of this could be the fewer validation slices or the different initial resolution of the Athens dataset.

VI. CONCLUSION: OPEN CHALLENGES AND FUTURE ACTIVITIES

The paper proposed federated and decentralized learning tools to support smart healthcare networks and medical diagnosis. An example of implementation was given in the context of brain tumor segmentation. Parameter server (PS) based federated learning (FL) and fully decentralized FL tools relying upon average consensus have been implemented on top of the MQTT transport protocol, while different network architectures and related designs were proposed to exploit synchronous and/or asynchronous coordination among the clients and the PS, when used. In particular, leveraging distinct levels of asynchronicity on the clients was found as a more efficient option in the presence of heterogeneous nodes and data, as letting the medical nodes to fully explore their local examples before publishing their updates. Asynchronous consensus is also a good-compromise between resource utilization and performances. The proposed architectures were extensively tested on medical data composed of heterogeneous datasets from public and private sources, demonstrating first of all the advantage of the FL system on the centralized training, and furthermore, the main benefits of the MQTT protocol when it comes to reliability, bandwidth efficiency and scalability.

The applications of FL for healthcare and automatic diagnosis are expected to quickly mature in the coming years targeting robust, scalable and privacy-preserving health services. In particular, distributed learning is expected to realize larger-scale and collaborative healthcare systems possibly opening to fully decentralized diagnosis operations, as opposed to centralized analytics on data centers. Besides the promising opportunities highlighted by the paper, future developments of FL are expected to address the robustness of classical and decentralized tools against adversarial manipulations and data poisoning. In addition, the trained models typically observed in medical applications have very large size, ranging from 30 and up to 150 MB [62] per medical node and learning round: when the FL process is implemented over wireless channels, the communication bottleneck should be carefully addressed, possibly via quantization, compression and model updates sparsification. As highlighted in the case study, excluding slow medical nodes (stragglers) from the federation is also not recommended in healthcare networks as these might be fundamental to improve model generalization. Trade-off solutions should be therefore investigated. A properly designed FL system can obtain performances meeting the needs of healthcare professionals. Nevertheless, a quantitative evaluation of trustworthiness metrics is also of fundamental importance to ensure the robustness of the FL platform.

ACKNOWLEDGMENT

Part of this work is performed using anonymized medical images from the 2nd Department of Radiology at Attikon University Hospital in Athens in the framework of a joint CERN/ Attikon University Hospital thesis. The authors would like to thank Prof E. Efstathopoulos and Prof. E. Karavasilis and their teams for their valuable support and guidance in the clinical interpretation of the results.

REFERENCES

[1] D. Wang, A. Khosla, R. Gargeya, H. Irshad, and A. H. Beck, "Deep learning for identifying metastatic breast cancer," Jun. 2016, *arXiv:1606.05718*.

[2] A. Ghubaish, T. Salman, M. Zolanvari, D. Unal, A. Al-Ali, and R. Jain, "Recent advances in the Internet-of-Medical-Things (IoMT) systems security," *IEEE Internet Things J.*, vol. 8, no. 11, pp. 8707–8718, Jun. 2021.

[3] C. Xu, N. Wang, L. Zhu, K. Sharif, and C. Zhang, "Achieving searchable and privacy-preserving data sharing for cloud-assisted E-healthcare system," *IEEE Internet Things J.*, vol. 6, no. 5, pp. 8345–8356, Oct. 2019.

[4] J. Konečný, B. McMahan, and D. Ramage, "Federated optimization: Distributed optimization beyond the datacenter," Nov. 2015, *arXiv:1511.03575*.

[5] N. Rieke, J. Hancox, W. Li, F. Milletarì, H. R. Roth, S. Albarqouni, S. Bakas, M. N. Galtier, B. A. Landman, K. Maier-Hein, S. Ourselin, M. Sheller, R. M. Summers, A. Trask, D. Xu, M. Baust, and M. J. Cardoso, "The future of digital health with federated learning," *npj Digit. Med.*, vol. 3, no. 1, pp. 1–7, Dec. 2020.

[6] M. J. Sheller, B. Edwards, G. A. Reina, J. Martin, S. Pati, A. Kotrotsou, M. Milchenko, W. Xu, D. Marcus, R. R. Colen, and S. Bakas, "Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data," *Sci. Rep.*, vol. 10, no. 1, p. 12598, Dec. 2020.

[7] S. Warnat-Herresthal, H. Schultze, K. L. Shastry, S. Manamohan, S. Mukherjee, V. Garg, R. Sarveswara, K. Händler, P. Pickkers, N. A. Aziz, and S. Ktena, "Swarm learning for decentralized and confidential clinical machine learning," *Nature*, vol. 594, no. 7862, pp. 265–270, Jun. 2021, doi: 10.1038/s41586-021-03583-3.

[8] H. B. McMahan, E. Moore, D. Ramage, and B. A. Y. Arcas, "Federated learning of deep networks using model averaging," Feb. 2016, *arXiv:1602.05629*.

[9] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, and H. V. Poor, "Federated learning meets blockchain in edge computing: Opportunities and challenges," *IEEE Internet Things J.*, vol. 8, no. 16, pp. 12806–12825, Aug. 2021.

[10] C. Ma, J. Li, M. Ding, H. Yang, F. Shu, T. Suek, and H. V. Poor, "On safeguarding privacy and security in the framework of federated learning," *IEEE Netw.*, vol. 34, no. 4, pp. 242–248, Jul./Aug. 2020.

[11] P. Kairouz *et al.*, "Advances and open problems in federated learning," 2019, *arXiv:1912.04977*.

[12] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," Nov. 2018, *arXiv:1811.03604*.

[13] M. J. Sheller, G. A. Reina, B. Edwards, J. Martin, and S. Bakas, "Multi-institutional deep learning modeling without sharing patient data: A feasibility study on brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*. Cham, Switzerland: Springer, Jan. 2019, pp. 92–104.

[14] S. Samarakoon, M. Bennis, W. Saad, and M. Debbah, "Distributed federated learning for ultra-reliable low-latency vehicular communications," *IEEE Trans. Commun.*, vol. 68, no. 2, pp. 1146–1159, Nov. 2020.

[15] S. Savazzi, M. Nicoli, M. Bennis, S. Kianoush, and L. Barbieri, "Opportunities of federated learning in connected, cooperative, and automated industrial systems," *IEEE Commun. Mag.*, vol. 59, no. 2, pp. 16–21, Feb. 2021.

[16] S. R. Pokhrel and J. Choi, "Federated learning with blockchain for autonomous vehicles: Analysis and design challenges," *IEEE Trans. Commun.*, vol. 68, no. 8, pp. 4734–4746, Aug. 2020.

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

[18] S. Savazzi, M. Nicoli, and V. Rampa, "Federated learning with cooperating devices: A consensus approach for massive IoT networks," *IEEE Internet Things J.*, vol. 7, no. 5, pp. 4641–4654, May 2020, doi: 10.1109/JIOT.2020.2964162.

[19] H. Wu and P. Wang, "Fast-convergent federated learning with adaptive weighting," *IEEE Trans. Cognit. Commun. Netw.*, vol. 7, no. 4, pp. 1078–1088, Dec. 2021.

[20] M. Havaei, F. Dutil, C. Pal, H. Larochelle, and P.-M. Jodoin, "A convolutional neural network approach to brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi, B. Menze, O. Maier, M. Reyes, and H. Handels, Eds. Cham, Switzerland: Springer, Jan. 2016, pp. 195–208.

[21] B. H. Menze *et al.*, "The multimodal brain tumor image segmentation benchmark (BRATS)," *IEEE Trans. Med. Imag.*, vol. 34, no. 10, pp. 1993–2024, Oct. 2015.

[22] S. Bakas, H. Akbari, A. Sotiras, M. Bilello, M. Rozycki, J. S. Kirby, J. B. Freymann, K. Farahani, and C. Davatzikos, "Advancing the cancer genome atlas glioma MRI collections with expert segmentation labels and radiomic features," *Sci. Data*, vol. 4, no. 1, Dec. 2017, Art. no. 170117.

[23] S. Bakas *et al.*, "Identifying the best machine learning algorithms for brain tumor segmentation, progression assessment, and overall survival prediction in the brats challenge," Mar. 2019, *arXiv:1811.02629*.

[24] A. Myronenko, "3D MRI brain tumor segmentation using autoencoder regularization," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi, S. Bakas, H. Kuijf, F. Keyvan, M. Reyes, and T. van Walsum, Eds. Cham, Switzerland: Springer, Jan. 2019, pp. 311–320.

[25] O. Ronneberger, P. Fischer, and T. Brox, "U-net: Convolutional networks for biomedical image segmentation," in *Medical Image Computing and Computer-Assisted Intervention—MICCAI*, N. Navab, J. Hornegger, W. M. Wells, and A. F. Frangi, Eds. Cham, Switzerland: Springer, Nov. 2015, pp. 234–241.

[26] Z. Jiang, C. Ding, M. Liu, and D. Tao, "Two-stage cascaded U-Net: 1st place solution to brats challenge 2019 segmentation task," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi and S. Bakas, Eds. Cham, Switzerland: Springer, May 2020, pp. 231–241.

[27] IntelAI. *UNet*. Accessed: Nov. 2021. [Online]. Available: https://github.com/IntelAI/unet/tree/master/2D

[28] M. A. Ferrag, O. Friha, L. Maglaras, H. Janicke, and L. Shu, "Federated deep learning for cyber security in the Internet of Things: Concepts, applications, and experimental analysis," *IEEE Access*, vol. 9, pp. 138509–138542, 2021.

[29] H. Kim, J. Park, M. Bennis, and S.-L. Kim, "Blockchained on-device federated learning," *IEEE Commun. Lett.*, vol. 24, no. 6, pp. 1279–1283, Jun. 2020.

[30] Z. Chen and E. G. Larsson, "Consensus-based distributed computation of link-based network metrics," *IEEE Signal Process. Lett.*, vol. 28, pp. 249–253, 2021.

[31] M. M. Amiri and D. Gündüz, "Machine learning at the wireless edge: Distributed stochastic gradient descent over-the-air," *IEEE Trans. Signal Process.*, vol. 68, pp. 2155–2169, Mar. 2020.

[32] M. Blot, D. Picard, M. Cord, and N. Thome, "Gossip training for deep learning," 2016, *arXiv:1611.09726*.

[33] Y. Qu, S. R. Pokhrel, S. Garg, L. Gao, and Y. Xiang, "A blockchained federated learning framework for cognitive computing in industry 4.0 networks," *IEEE Trans. Ind. Informat.*, vol. 17, no. 4, pp. 2964–2973, Apr. 2021.

[34] L. Barbieri, S. Savazzi, M. Brambilla, and M. Nicoli, "Decentralized federated learning for extended sensing in 6G connected vehicles," *Veh. Commun.*, vol. 33, Jan. 2022, Art. no. 100396. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S2214209621000656

[35] A. Guha Roy, S. Siddiqui, S. Pölsterl, N. Navab, and C. Wachinger, "BrainTorrent: A peer-to-peer environment for decentralized federated learning," May 2019, *arXiv:1905.06731*.

[36] *Federated Learning Brings AI With Privacy to Hospitals*. Accessed: Nov. 2021. [Online]. Available: https://healthcare-in-europe.com/en/news/federated-learning-brings-ai-with-privacy-to-hospitals.html

[37] D. Yang, Z. Xu, W. Li, A. Myronenko, H. R. Roth, S. Harmon, S. Xu, B. Turkbey, E. Turkbey, X. Wang, W. Zhu, G. Carrafiello, F. Patella, M. Cariati, H. Obinata, H. Mori, K. Tamura, P. An, B. J. Wood, and D. Xu, "Federated semi-supervised learning for COVID region segmentation in chest CT using multi-national data from China, Italy, Japan," *Med. Image Anal.*, vol. 70, May 2021, Art. no. 101992.

[38] J. Hoydis, F. A. Aoudia, A. Valcarce, and H. Viswanathan, "Toward a 6G AI-native air interface," *IEEE Commun. Mag.*, vol. 59, no. 5, pp. 76–81, May 2021.

[39] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Process. Mag.*, vol. 37, no. 3, pp. 50–60, May 2020, doi: 10.1109/MSP.2020.2975749.

[40] L. Bottou, "Online algorithms and stochastic approximations," in *Online Learning and Neural Networks*, D. Saad, Ed. Cambridge, U.K.: Cambridge Univ. Press, Oct. 2012. [Online]. Available: http://leon.bottou.org/papers/bottou-98x

[41] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. Y. Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proc. AISTATS*, Apr. 2017, pp. 1273–1282.

[42] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," May 2015, *arXiv:1412.6980*.

[43] M. G. Arivazhagan, V. Aggarwal, A. K. Singh, and S. Choudhary, "Federated learning with personalization layers," Dec. 2019, *arXiv:1912.00818*.

[44] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," in *Proc. 3rd Int. Conf. Learn. Represent. (ICLR)*, Y. Bengio and Y. LeCun, Eds., San Diego, CA, USA, May 2015.

[45] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," in *Proc. IEEE Conf. Comput. Vis. Pattern Recognit. (CVPR)*, Feb. 2016, pp. 770–778.

[46] D. Cheng and E. Y. Lam, "Transfer learning U-Net deep learning for lung ultrasound segmentation," Oct. 2021, *arXiv:2110.02196*.

[47] G. Soatti, M. Nicoli, S. Savazzi, and U. Spagnolini, "Consensus-based algorithms for distributed network-state estimation and localization," *IEEE Trans. Signal Inf. Process. Netw.*, vol. 3, no. 2, pp. 430–444, Mar. 2017.

[48] M. Blot, D. Picard, M. Cord, and N. Thome, "Gossip training for deep learning," Nov. 2019, *arXiv:1611.09726*.

[49] A. Lalitha, O. C. Kilinc, T. Javidi, and F. Koushanfar, "Peer-to-peer federated learning on graphs," Jan. 2019, *arXiv:1901.11173*.

[50] W. Li, F. Milletarì, D. Xu, N. Rieke, J. Hancox, W. Zhu, M. Baust, Y. Cheng, S. Ourselin, M. J. Cardoso, and A. Feng, "Privacy-preserving federated brain tumour segmentation," in *Machine Learning in Medical Imaging*. Cham, Switzerland: Springer, Oct. 2019, pp. 133–141.

[51] R. Shokri and V. Shmatikov, "Privacy-preserving deep learning," in *Proc. 53rd Annu. Allerton Conf. Commun., Control, Comput. (Allerton)*, Sep. 2015, pp. 909–910.

[52] D. Połap, G. Srivastava, A. Jolfaei, and R. M. Parizi, "Blockchain technology and neural networks for the Internet of Medical Things," in *Proc. IEEE Conf. Comput. Commun. Workshops (INFOCOM WKSHPS)*, Jul. 2020, pp. 508–513.

[53] M. Aledhari, R. Razzak, R. M. Parizi, and F. Saeed, "Federated learning: A survey on enabling technologies, protocols, and applications," *IEEE Access*, vol. 8, pp. 140699–140725, 2020.

[54] C. H. Sudre, W. Li, T. Vercauteren, S. Ourselin, and M. J. Cardoso, "Generalised dice overlap as a deep learning loss function for highly unbalanced segmentations," in *Proc. Int. Workshop Deep Learn. Med. Image Anal.*, in Lecture Notes in Computer Science, Sep. 2017, pp. 240–248, doi: 10.1007/978-3-319-67558-9_28.

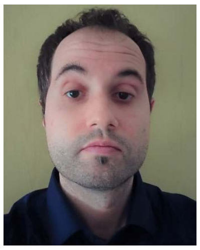
[55] L. R. Dice, "Measures of the amount of ecologic association between species," *Ecology*, vol. 26, no. 3, pp. 297–302, Jul. 1945.

[56] *MQTT V3.1 Protocol Specification*. Accessed: Nov. 2021. [Online]. Available: https://mqtt.org

[57] J. Dizdarević, F. Carpio, A. Jukan, and X. Masip-Bruin, "A survey of communication protocols for Internet of Things and related challenges of fog and cloud computing integration," *ACM Comput. Surv.*, vol. 51, no. 6, pp. 1–29, Jan. 2019, doi: 10.1145/3292674.

[58] *Pickle vs JSON—Which is Faster in Python3?* Accessed: May 4, 2020. [Online]. Available: https://tinyurl.com/3yznvvy3
[59] S. Lee, H. Kim, D.-K. Hong, and H. Ju, "Correlation analysis of MQTT loss and delay according to QoS level," in *Proc. Int. Conf. Inf. Netw. (ICOIN)*, Jan. 2013, pp. 714–717.
[60] (2021). *Software Repository*. Accessed: Sep. 2021. [Online]. Available: https://github.com/BernardoCama/FL_MQTT
[61] X. Zhu, K. Guo, S. Ren, B. Hu, M. Hu, and H. Fang, "Lightweight image super-resolution with expectation-maximization attention mechanism," *IEEE Trans. Circuits Syst. Video Technol.*, early access, May 10, 2021, doi: 10.1109/TCSVT.2021.3078436.
[62] F. Isensee, P. F. Jäger, P. M. Full, P. Vollmuth, and K. H. Maier-Hein, "nnU-Net for brain tumor segmentation," in *Brainlesion: Glioma, Multiple Sclerosis, Stroke and Traumatic Brain Injuries*, A. Crimi and S. Bakas, Eds. Cham, Switzerland: Springer, Mar. 2021, pp. 118–132.

BERNARDO CAMAJORI TEDESCHINI (Graduate Student Member, IEEE) received the bachelor's degree (Hons.) in computer science and the master's degree (Hons.) in telecommunications, specializing in communication networks and internet from the Politecnico di Milano, in July 2019 and October 2021, respectively, where he is currently pursuing the Ph.D. degree in information technology with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB). His research interests include federated learning, machine learning, and localization methods.



STEFANO SAVAZZI (Member, IEEE) received the M.Sc. and Ph.D. degrees (Hons.) in ICT from the Politecnico di Milano, Italy, in 2004 and 2008, respectively. In 2012, he joined the Institute of Electronics, Computer and Telecommunication Engineering (IEIIT), Consiglio Nazionale delle Ricerche (CNR), as a Researcher. He was a Visiting Researcher with Uppsala University, in 2005, and the University of California at San Diego, in 2007. He has coauthored over 100 scientific publications (Scopus). His current research interests include distributed signal processing, machine learning and networking aspects for the Internet of Things, radio localization, and vision technologies. He won the Dimitris N. Chorafas Foundation Award, in 2008. He is serving as an Associate Editor for *Frontiers in Communications and Networks* and *Wireless Communications and Mobile Computing* and Lead Guest Editor for the Special Issue on Radio Sensing and Sensor Networks in Sensors (MDPI).

ROMAN STOKLASA received the M.Sc. (Hons.) and Ph.D. degrees in informatics, image processing, computer vision and machine learning from the Faculty of Informatics, Masaryk University, Brno, Czech Republic, in 2010 and 2016, respectively. In 2013, he was on a visiting internship position with the School of Digital Media Technology, Birmingham City University, U.K. Since 2016, he has been working as a Postdoctoral Researcher with the Centre for Biomedical Image Analysis, Masaryk University. Since 2020, he has been a Senior Fellow with the Technology Department, CERN, Geneva, Switzerland, working on projects related to data and image analysis, machine learning, and predictive maintenance. His research interests include deep learning, image and signal processing, medical and biomedical image analysis, computer vision, and image perception.

LUCA BARBIERI (Graduate Student Member, IEEE) received the B.Sc. and M.Sc. *(cum laude)* degrees in telecommunication engineering from the Politecnico di Milano, in 2017 and 2019, respectively, where he is currently pursuing the Ph.D. degree in information technology with the Dipartimento di Elettronica, Informazione e Bioingegneria (DEIB). His current research interests include machine learning and localization techniques for vehicular and industrial networks.

IOANNIS STATHOPOULOS received the degree in physics and the master's degree in medical physics from the University of Patras. He is currently pursuing the Ph.D. degree with the Department of Technology, European Organization for Nuclear Physics (CERN) and the Medical School of National and the Kapodistrian University of Athens. The subject of his dissertation is the use of machine learning algorithms for detection and classification of brain abnormalities from magnetic resonance images. His research interests include the usage of artificial intelligence, machine and deep learning methods to support medical diagnosis, and pattern recognition analysis using heterogeneous medical data and medical image analysis.

MONICA NICOLI (Member, IEEE) received the M.Sc. (Hons.) and Ph.D. degrees in communication engineering from the Politecnico di Milano, Milan, Italy, in 1998 and 2002, respectively. She was a Visiting Researcher with ENI Agip, from 1998 to 1999, and Uppsala University, in 2001. In 2002, she joined the Politecnico di Milano as a Faculty Member. She is currently an Associate Professor in telecommunications with the Department of Management, Economics and Industrial Engineering. She has coauthored around 150 scientific publications. Her research interests include signal processing, machine learning, and wireless communications, with emphasis on smart mobility and the Internet of Things (IoT) applications. She was a recipient of the Marisa Bellisario Award, in 1999, and a co-recipient of the best paper awards of the IEEE Symposium on Joint Communications and Sensing, in 2021, the IEEE Statistical Signal Processing Workshop, in 2018, and the *IET Intelligent Transport Systems* journal, in 2014. She has served as an Associate Editor for the *EURASIP Journal on Wireless Communications and Networking*, from 2010 to 2017, and a Lead Guest Editor for the Special Issue on Localization in Mobile Wireless and Sensor Networks, in 2011. She is also an Associate Editor of the IEEE TRANSACTIONS ON INTELLIGENT TRANSPORTATION SYSTEMS.

LUIGI SERIO received the M.Sc. degree in nuclear engineering from the Politecnico di Milano, Italy, and the Ph.D. degree in mechanical engineering from Cranfield University, U.K. He is currently a Senior Staff, a Principal Investigator, and the Project Leader with the Technology Department, CERN, Geneva, Switzerland. He is also a Chartered Nuclear Engineer, a Radiation Protection Adviser, and a PMP. He has published more than 100 papers in the field of nuclear and cryogenic engineering and technology. He has an international patent and two theses. His field of expertise is in cryogenics, mechanical, instrumentation, nuclear and thermo-hydraulic engineering, availability and reliability, machine learning and artificial intelligence. He has also worked and collaborated as an Advisor and a Review Committee Member with international projects in the field of nuclear fusion and particle accelerators at JET Joint Undertaking, U.K., SLAC, Stanford, USA, FERMILAB, USA, and ITER Organization, Cadarache, France.