An Adaptive Model Averaging Procedure for Federated Learning (AdaFed)

Alessandro Giuseppi, Lucrezia Della Torre, Danilo Menegatti, Francesco Delli Priscoli, and Antonio Pietrabissa
University of Rome La Sapienza, Rome, Italy
Email: {giuseppi, dellatorre, menegatti, dellipriscoli, pietrabissa}@diag.uniroma1.it

Cecilia Poli
Istituto Superiore di Sanità, Rome, Italy
Email: cecilia.poli@iss.it

***Abstract*—Federated Learning (FL) is an enabling technology for Machine Learning in scenarios in which it is impossible, for privacy and/or regulatory reasons, to analyze data in a centralized manner. FL envisages that distributed clients cooperate to learn a model without any data exchange, in favor of a model averaging procedure that is coordinated by a server. In this work, we present the Adaptive Federated Learning (AdaFed) algorithm, that extends the original Federated Averaging algorithm by: (i) dynamically weighting the local models, based on their performance, for the averaging procedure; (ii) adapting the loss function at every communication round depending on the training behavior. This work specializes AdaFed for both classification and regression tasks, and reports several validation tests on benchmarking dataset, showing its enhanced robustness against unbalanced data distributions and adversarial clients.**

***Index Terms*—federated learning, distributed learning systems, adaptive learning, deep neural networks**

I. Introduction

Federated Learning (FL) is a distributed learning solution to address Machine Learning (ML) problems without the need of collecting the available data in a single data center. FL finds application in scenarios where data are distributed across a multitude of sources that, due to privacy or communication constraints, cannot share it with each other or with a centralised entity. The need of analysing data locally becomes paramount when dealing with personal data (e.g., collected by smartphones) or sensitive information (e.g., business data).

Hence, heavily regulated fields such as healthcare [1], and connecting the fragmented data sources while preserving privacy [2], could benefit from the application of FL. For example, in the medical field, the use of deep learning methodologies in the identification of complex models, whose potential is emphasized in the literature with numerous applications in radiology, pathology, and genomics [3], is severely limited by both the inability of individual institutions to have sufficiently large and diverse datasets, and difficulties related to data privacy and ownership, in the case of multi-institutional collaborations based on centrally-share patient data, especially in international collaborations, and are also not suitable for cases where there are the number of institutions is large. As a result, the knowledge generated worldwide remains distributed among multiple institutions, increasing the need to look for alternative approaches: because of its privacy preserving characteristics, FL represents a collaborative learning approach enabling multi-institutional collaborative learning tasks for intelligent healthcare applications. In other words, given a network of hospitals, FL makes it possible to share the knowledge gained from the model of each hospital to improve that of the whole federation, without having to collect data from all hospitals in a single server. This solution can be made GDPR compliant.

In this work, we propose the Adaptive Federated Learning algorithm, AdaFed, which, although designed to extend the formulation and results of the original FL algorithm Federated Averaging (FedAvg), the concepts behind its innovations are independent of the specific implementation and may be seamlessly translated to other algorithms such as FedProx [4]. Specifically, in FL the server's model is updated at every communication round by averaging the models of the federated clients, trained on their locally available data, while AdaFed proposes a two-step procedure to improve both the model averaging and the local training processes by i) dynamically weighting the client models contributions to the federation based on their performance, and ii) adapting the federated loss function depending the global model performance at each communication round.

The evaluation of AdaFed in terms of versatility, performance improvements and capability of addressing new challenging scenarios is illustrated by comparison with FedAvg on different tasks involving several different data-sets.

The rest of this paper reports in Section II a brief state-of-the-art, while in Section III the AdaFed algorithm is presented for both classification and regression tasks.

Manuscript received March 16, 2022; revised May 13, 2022; accepted June 16, 2022.

Numerical tests are used for validation of the algorithm and reported in Section IV, and, finally, Section V highlights possible future works and concludes the paper.

II. Related Works and Main Contributions

The concept of FL was introduced in 2016 by the authors of [7] and, in its original formulation [6], [7], it was presented as a solution to specifically address the collaboration among a group of smartphones by means of an iterative model-averaging procedure. According to which, local model updates, independently computed by the smartphones, were gathered and averaged by a centralised server that then propagated the updated *global* model in the network, as depicted in Fig. 1.

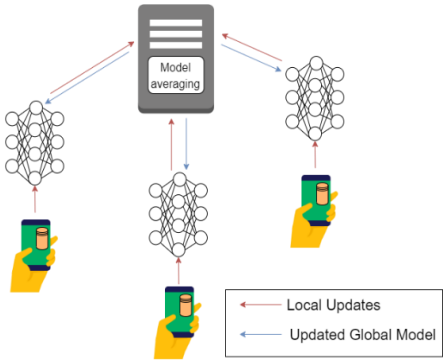

Figure 1. Sketch of the architecture for "Federated Learning" [5], as originally envisaged by the FedAvg algorithm presented in [6].

As examined within the later studies [5], [8], [9], since its introduction, FL was extended to different architectures and uses cases, with several works focusing on improving its privacy preserving and security related features [10]-[13] to prevent direct or indirect data leakage [14], and reducing communication costs related to the distributed training [15]-[17]. FL found application in several domains, ranging from smartphones/Internet of Things tasks such as Natural Language Processing (NLP) [18]-[20], image analysis [21], [22] and distributed sensing and computing [9], [23], to scenarios where organisations and institutions cooperate to achieve better models to analyse complex and highly confidential data, typical of the healthcare domain [24]-[26]. Contrary to federated database systems, where data can be freely distributed by a central entity, the typical scenario for a FL application is characterised by the need to analyse data partitioned *as given*, implying that FL algorithms need to be able to cope with data that are:

- Non-IID and imbalanced, as the geographical topology of the federated organisations or data sources may significantly affect the data distribution and collecting procedures;
- Extremely distributed, namely when the quantity of individual data samples is lower than the number of connected devices.

The fundamental idea at the basis of this work is to assign to the various clients a different weight for the averaging, based on their contribution, in terms of performance and attained knowledge, to the federation. Similar concepts have been explored by other recent works, such as done by the authors of the FOCUS algorithm [27], that designed a procedure to determine a weighting strategy for the various clients that is based on a so-called "credibility score". Such score was designed to reduce the sensitivity of the model trained by the federation to un-even levels of labelling quality of the clients' data. This allows to increase the robustness of the training algorithm to both different data harvesting/collection procedures and also human factors. The labelling quality is inferred by the FOCUS algorithm by correlating the performance of the averaged model on the distributed datasets with the performance of the clients' models on the server dataset-, whose labelling quality is assumed to be high). The AdaFed algorithm was not specifically designed to provide robustness to labelling quality disparity, but instead it focuses on evaluating the quality of the clients' models and their consequent contribution to the federation knowledge discovery process. In fact, AdaFed allows FL solutions to cope with extremely unbalanced scenarios with un-even data distributions and, also, allows to deal with the presence of compromised, or even malicious, clients.

The second main contribution AdaFed is the inclusion of an *adaptive loss functions* [28]-[31] into the federated training process. Advanced loss functions are commonly found in several complex ML applications, such as computer vision [28], [31], [32] and as multi-objective learning, where Pareto-like optimality [29] is sought by minimizing multiple competing objectives.

Having the loss function evolve during and adapt during the training process, depending on the model performance evaluated during the training itself, opens up some interesting and still not completely investigated possibilities. Due to the iterative nature of the FL process, it is seamless to update the loss function at the end of each communication round (i.e., immediately after the model averaging).

The main innovations of the present work are:

1) The proposal of a new federated learning algorithm, namely AdaFed, that features two functionalities to improve the training process:
   a) The dynamic weighting of the clients' models in the model averaging procedure, based on their performance.
   b) The introduction of an adaptive loss function in the FL setting.
2) the validation of AdaFed on multiple scenarios, involving convolutional neural networks, transfer learning and compromised/malicious federations.

III. AdaFed: Adaptive Federated Learning

The AdaFed algorithm is reported in the Algorithm 1 table, where the two main innovations of the algorithm are highlighted. Namely, the key functionalities introduced by AdaFed are:

1) **Weighted Model Averaging** (lines 4-9): When the server carries-out the model averaging procedure and it gathers the clients' models, a preliminary evaluation of all the models is conducted on a dataset available to the server. According to the models' performances, a different weight is given

to each model for the model averaging, so that better performing models are given more focus.

Algorithm 1. AdaFed: Performance-Based Adaptive Federated Learning algorithm

1: **SERVER'S UPDATE:**
2: **for** each round $t = 1, 2, \ldots, R$ **do**
3: ClientsUpdate
4: **for** each client $i = 1, 2, \ldots, K$ **do**
5: receive the client's model w_i
6: evaluate w_i on the server test set
7: use the evaluation to determine the weight p_i
8: **end for**
9: update the server's model $w_S \leftarrow \frac{\sum_{i=1}^{n_c} w_i p_i}{\sum_{i=1}^{n_c} p_i}$
10: evaluate its performance p_S on the server test set
11: adapt the loss function l depending on p_S
12: propagate w_S and l to the clients
13: **end for**

14: **ClientsUpdate**:
15: **for** each client $i = 1, 2, \ldots, K$ **do**
16: $w_i \leftarrow w_S$
17: **for** each local epoch $j = 1, 2, \ldots, E$ **do**
18: **for** each mini-batch b of size B **do**
19: $w_i \leftarrow w_i - \eta \nabla l(w, b)$
20: **end for**
21: **end for**
22: return w_i to server
23: **end for**

The performance can be evaluated, generally speaking, with any indicator that captures the quality of the solution proposed by the model for the given task (e.g., accuracy for classification tasks).

2) **Adaptive Loss** (lines 10-12): the server sends to the clients the averaged model, as in standard FL approaches, and sets a new loss function that was updated based on the performance archived by the new averaged model, according to a use-case dependant indicator (e.g., update class weights for classification tasks depending on recall).

For the sake of clarity, Algorithm 1 is reported using a similar syntax to the one used for FedAvg in [6]. In general, the privacy enhancing characteristics and features developed for more recent and specialized FL algorithms may be included in the AdaFed formulation, provided that they comply with the standard iterative model averaging structure of the original FedAvg.

As in [27], AdaFed estimates the contribution to the federation of the various clients in order to determine their relative weights for the model averaging procedure.

The main limitation of AdaFed is that, for such an evaluation, it requires the availability of a test that is representative for the considered task.

The main advantages offered by AdaFed are that, by combining the dynamic weighted model averaging procedure with an adaptive loss function, it makes so: i) more focus is given to better performing clients, while reducing or preventing the negative effects caused by compromised and malicious clients and ii) more attention is given to data samples and/or features required to improve the model performance, such as in the case of the most rare/harder to discern labels and characteristics in classification tasks.

A. Application: AdaFed for Classification Tasks

Multi-class classification is a typical example found in computer vision and in general Machine Learning applications. The most common loss function for this kind of problems is the so-called categorical cross-entropy:

$$l(X, Y) = -\frac{1}{M}\sum_{c=1}^{C}\sum_{m=1}^{M} y_m^c \log(\hat{y}_m^c) \quad (1)$$

where $x_m \in X$ and $y_m \in Y$ are the m-th data sample and label, respectively, in the dataset (X, Y), M and C are respectively the number of data samples and classes, y_m^c and y_m^c denote the c-th component of the vectors y_m and $\hat{y}_m$ and are respectively the true and predicted labels for the sample m regarding class c. Note that $\hat{y}_m^c$ is typically produced, for single-label problems, by a deep neural network with a *softmax* output activation function and can be interpreted as the probability of correctness for the given label.

Note that, in the standard formulation, the categorical cross-entropy does not compensate for imbalanced class distributions, that in turn characterize several FL scenarios. In order to ease the learning when dealing with unbalanced data distributions is to utilise a weighted categorical cross-entropy, that is:

$$l(X, Y) = -\frac{1}{M}\sum_{c=1}^{C}\sum_{m=1}^{M} \kappa^c y_m^c \log(\hat{y}_m^c) \quad (2)$$

where κ^c is a class-dependent weight. Regarding such class weights, works such as [33] set them as proportional to the inverse class numerosity available in the training set, while works such as [34] employ a complex weight that is derived from the class recognition complexity.

In the same direction of [28], this paper does not consider any rule-based policy to set the loss parameters, as it is commonly done by most approaches in the literature, and instead [28] dynamically modifies the loss structure/parameters during the training.

Following the approach of [28], [33], [34], we choose for our testing the class dependent weight κ^c to be inversely proportional to the performance p_S obtained by the server model on the server test set with respect to its corresponding class, expressed by means of its F_1^c-score, e.g., $\kappa^c = 1/(F_1^c + \epsilon)$, where $\epsilon < 1$ is a design parameter

to limit κ^c. Note that this choice is however arbitrary as other metrics to evaluate p_S can be chosen, as shown in Section IV. The rationale behind this adaptive loss logic is to encourage the clients to focus, during their training phase, on classes which are misclassified by the global model.

Referring to Algorithm 1, this choice translates in having $p_S = \{F_1^c, \text{ for } c = 1, \dots, C\}$ (line 10) and updating the loss function l with the new weights κ^c derived from the F_1^c-scores (line 11).

The same idea is used in the Weighted Model Average step. Note that, in principle, the weight p_i of the *i*-th client can be computed via the performance of its model over the server test set with a different metric than the one used for the Adaptive Loss step, so instead of the F_1-score one may utilise for example the model accuracy or the diagnostic odds ratio, depending on the specific use case.

B. Application 2: AdaFed for Regression Tasks

As introduced, AdaFed can be seamlessly applied to regression tasks provided some awareness on the considered task and overall FL design. For example, in regression problems one may have that an application-dependent metric, not directly related to the loss, captures the performance of the model at solving the considered task. For instance, in our example (see Simulation 3), we have that the task objective is to estimate the number of cells in a given microscope photo, so we will consider the mean absolute percentage error on the counting of cells, whereas the models' loss will be related to the mean squared error between a target image and the image generated by the deep neural network.

Furthermore, it is in principle possible to consider a loss function with a variable structure such as the one proposed in [28] and adapt its hyper parameters at each communication round, depending on the server's model performance evaluated on an ad-hoc dataset.

C. Limitations

The main limitation of the proposed framework is the availability of a server test set that is qualitatively and quantitatively representative of the learning task. While its numerosity is not required to be particularly high, on certain scenarios it may not be possible to assume the existence of such a set. A possible solution would be to share the local models in the network and collect their performance on the distributed test set available to the various clients. By properly averaging the collected performance it is then possible to determine the various performance weights p_i and obtain the updated global model. This procedure may be iterated again for adapting the loss function l. Note that this approach, besides adding a significant communication overhead, may be sensitive to adversarial attacks as a level of trust in the evaluations of the clients is required if no adversary detection strategy is implemented. An alternative approach could be imposing to every member of the federation to share a small portion of their data, selected randomly, so that the resulting dataset may be representative for all the data available to the federation.

IV. Experiments

In this section, AdaFed is tested and compared to FedAvg in different scenarios. This section expands the results reported in [35] by considering more datasets and scenarios. The rationale behind all of the conducted tests is to evaluate how AdaFed performs in challenging settings to better capture the contribution of its innovative features. For this reason, AdaFed will be compared with FedAvg, as it is the baseline for FL solutions. In the remainder of the section, for the sake of presentation clarity, we will assume all the clients to send their model to the server at each communication round.

A. Simulation 1 - MNIST and OARF Classification Tasks

The first simulation considered deals with the MNIST [36] dataset. Such a simple dataset allows to better evaluate the effect of each of the proposed enhancements individually, provided AdaFed is deployed on an ad-hoc scenario.

We recall that the MNIST dataset consists in a set of 60k+10k labelled images of handwritten digits (from 0 to 9), and represents on of the most common baselines for classification tasks.

Simulation 1.A will focus on the Weighted Model Averaging step, while Simulation 1.B was designed to evaluate the Adaptive Loss functionality and, finally, Simulation 1.C validates the robustness AdaFed in the presence of adversarial clients.

In all three of such simulations, we set the parameters of AdaFed (refer to Algorithm 1) as $E = 5$ (number of epochs in the clients' update), $B = 100$ (batch size) and $R = 20$ (number of communication rounds). The server test set is composed by the whole MNIST test set, whereas clients' data are distributed as described in the following sections.

1) Simulation 1.A (MNIST) - Validation of weighted model averaging

For this simulation, we consider $K = 6$ clients, with five of them having access to 5500 samples of only two digits. The only exception is related to simulating the scarcity of samples for classes 0 and 5, for which we used only 100 and 200 samples respectively. The last client was instead assumed to have a small amount of data (500 samples) from each of the 10 classes.

For the training we used the standard architecture provided by Keras [37], that is composed by a stack of two convolutional layers and two fully-connected layers. The performance weight p_i of the *i*-th client for AdaFed is computed as its accuracy times the number of its available data samples, i.e., $p_i = accuracy_i \times \#training\_data_i$.

Fig. 2 shows that AdaFed starts with a higher accuracy from the very first communication round, as the weighted model averaging reduces the contribution of models that overfitted on the local data, thus converging faster than FedAvg. The presence of a client (client 6) that has access to data from all classes, even if in limited quantities, allows it to attain better a performance from the very start of the training when compared to the other clients that have only data from two classes at their disposal. Consequently,

AdaFed, thanks to its adaptive weighting strategy, gives more focus and weight to client 6 partially avoiding the negative effects caused by the other clients' data distributions. This advantage carries out during the entirety of the training, with FedAvg catching up only after over 12 communication rounds.

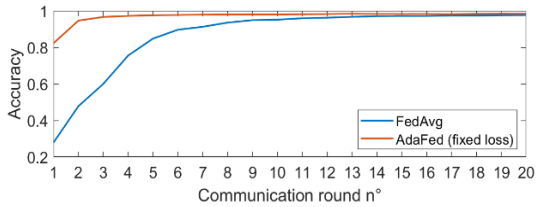

Figure 2. Simulation 1.A (MNIST): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

2) Simulation 1.B (MNIST) - Validation of adaptive loss function

In order to evaluate the contribution of the adapting loss function, we consider now a scenario with data that is now distributed in a less uniform way, as reported in Table I.

TABLE I. SIMULATION 1.B - DATA DISTRIBUTION AMONG CLIENTS

Client	Classes									
1	10	0	30	10	30	50	20	20	10	10
2	10	0	0	500	100	0	0	500	100	500
3	0	0	30	500	100	150	500	0	0	500
4	0	0	30	0	100	0	500	500	100	0
5	0	10	30	500	0	0	500	500	0	500
6	0	10	10	10	10	100	10	0	10	3000

We consider a weighted categorical cross-entropy loss, that is updated at the end of the various communication rounds depending on the F_1 -scores attained by the averaged model for the ten classes. In particular, the weights κ^c are set to be equal to $1/(F_1^c + 0.1)$. The logic behind this choice is to increase (10 times) the impact on the loss value for samples that belong to the commonly miss-classified (i.e., $F_1^c \to 0$) classes, whereas better recognised classes (i.e., $F_1^c \to 0$) are given lower focus.

Fig. 3 reports the accuracy, over the communication rounds, of FedAvg against two different AdaFed federations, one implementing both the weighted averaging and the adaptive loss procedure, and another that only implements the weighted model averaging as in the previous simulation, with $p_i = accuracy_i$.

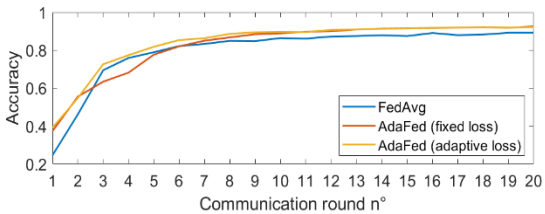

Figure 3. Simulation 1.B (MNIST): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

AdaFed is able to offset FedAvg by about 3% in both of its implementations, with the one including the adaptive loss converging faster than the other. Fig. 4 reports the Macro F_1-scores (i.e., the average of all the F_1^c-scores) and it shows that the adaptive loss implementation of AdaFed is able to better discern classes for which limited samples are available. This performance gap is highlighted in Fig. 5, where it can be seen that the F_1^c-scores increase more uniformly and faster when the adaptive loss is implemented.

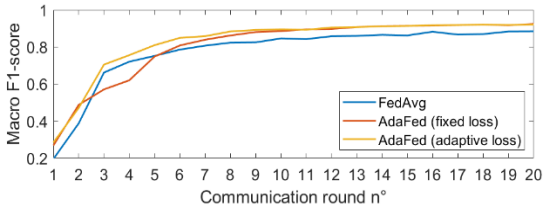

Figure 4. Simulation 1.B (MNIST): Evolution of the server model macro F_1-score over communication rounds, evaluated on the separated test set of the server.

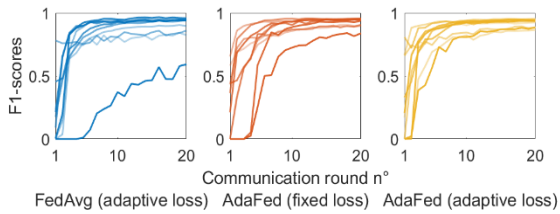

Figure 5. Simulation 1.B (MNIST): Evolution of the server model F_1-scores over communication rounds, evaluated on the separated test set of the server.

3) Simulation 1.C (MNIST) - Robustness to adversarial actors

Considering the data distribution of the previous simulation, we now focus on the resiliency of an AdaFed federation against malicious clients that aim at compromising the federation training. To this end, we consider two additional clients, 7 and 8 (i.e., $K = 8$), that share the same data distribution of clients 3 and 4 but with incorrect labeling for respectively 50% and 100% of their samples. Additionally, the new clients do not use in their training the averaged model received from the server and instead maintain their local model over the entire training.

Fig. 6 and Fig. 7 report that AdaFed (that implements both adaptive loss and weighted averaging functionalities) is unaffected by the adversarial clients (the difference with respect to the previous simulation is about 0.01% in accuracy and 0.015 in the macro F_1-score), while FedAvg performances are significantly impacted, with accuracy losing about 2% and the macro F_1-score being lower by about 0.1. We note that, in more complex scenarios, AdaFed weights p_i may be set using arbitrary logics (e.g., a quadratic or cubic function of the accuracy) to further decrease or annihilate the weights given to underperforming (and hence potentially compromised) clients, as will be demonstrated in the following simulation.

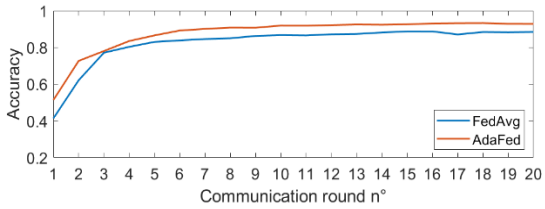

Figure 6. Simulation 1.C (MNIST): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

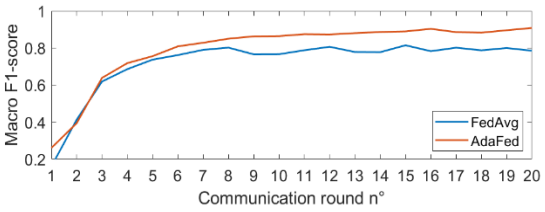

Figure 7. Simulation 1.C (MNIST): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

4) Simulation 1.D (OARF) - Robustness unfavorable data distributions and/or adversarial actors

Simulation 1.D considers a binary sentiment analysis classification task from the benchmark suite OARF [38], further highlighting the capabilities of AdaFed to cope with poorly distributed data. The parameters were set as $E = 3$, $B = 100$ and $R = 5$.

The results obtained for Simulation 1.D are applicable also in scenarios in which the data are distributed in an extremely unfavorable way. In this simulation, we consider the sentiment analysis dataset from the benchmark suite Open Application Repository for Federated Learning (OARF) [38], consisting in 50k entries from the Amazon Movie review dataset proposed in [39] for a binary classification task. We assume the data to be partitioned among $K = 5$ clients, as follows: 1 client only has access to ~ 5k samples from the negative class (i.e., negative review), 2 clients have access to a total of ~ 15k samples from the positive class (i.e., positive review), while the remaining two clients have ~ 10k and ~ 6k samples equally distributed over the two classes. Each agent employs a two-layer Long-Short-Term-Memory (LSTM) [40] neural network, with the same characteristics and data preprocessing employed in [38] and made available in [41]. The particular distribution of data makes so that all the clients that only have a single class available will tend to not generalise correctly, as always predicting the same class (even without considering the input data) will minimize their loss and let them reach a 100% accuracy. This in turn implies that their contribution to the federation will be questionable, if not negative, as the model they will forward for the averaging procedure will tend to be insensible to the input. To address such scenario, we propose for this simulation a different rule to compute the model weights p_i for the averaging procedure. Namely, we determine the weights as the accuracy percentage exceeding 55%, meaning that random models that reach an accuracy level of about 50% are associated to a 0 weight.

Fig. 8 reports a comparison of the accuracies attained by AdaFed and FedAvg in the described scenario, evaluated on the entire test set of the Amazon database. It is possible to note that AdaFed (orange) reaches an overall accuracy of 80% from the very first iterations, in line with [38], while FedAvg (blue) fails to converge and overs around 50%, which is the accuracy level expected from a random agent.

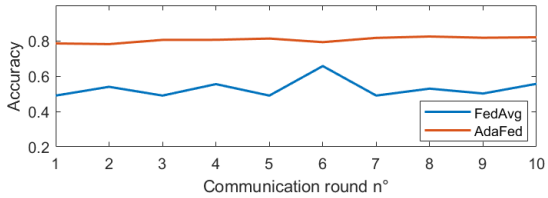

Figure 8. Simulation 1.D (OARF Sentiment Analysis): Evolution of the server model Macro F_1-score over communication rounds, evaluated on the separated test set of the server.

B. Simulation 2 - CIFAR10 and Transfer Learning

Having validated the effect of the two proposed modifications, we now test AdaFed on a more complex classification problem. The dataset considered for this simulation is the well-known CIFAR10 (Canadian Institute for Advanced Research) [42], and to further show the flexibility of the proposed approach we consider in this section a federated Transfer Learning [43], [44] solution. This case study is of particular importance, as transfer learning significantly reduces the number of trainable parameters by starting the training process with a neural network that was already trained to solve a similar task. The usage of such a network as the basis for the new predictor, effectively allows for a knowledge transfer between the previously solved problem and the new one (e.g., a typical solution in specialised computer vision task is to employ transfer learning with a general purpose image classifier that was trained on a complex - yet general - dataset such as ImageNet [45]).

The reduction of the number of the trainable parameter leads directly impacts the training complexity and communication overhead needed to sustain the federation, making the combination of FL and transfer learning an efficient and effective solution.

We also utilise this simulation to test AdaFed on a more complex task in a more general scenario that was not designed to stress any particular feature of the proposed algorithm.

Being the dataset constituted by ~ 60k 32×32 color images of 10 different classes, we consider $K = 8$ clients divided in two groups: the first four clients were given between 300 and 600 samples for each of the ten classes, while the remaining four only had data from five classes. The transfer learning model was constituted by the VGG19 [46] network trained for ImageNet [45] and attached to four dense layers with respectively 2014, 512, 256 and 128 neurons and ReLu [47] activation functions. The same adaptive loss and weighted averaging procedures as in Simulation 1 were employed. We also set $E = 5$, $B = 100$, $R = 20$.

Looking at Fig. 9, it is clear that AdaFed outperforms FedAvg starting from the 4th round, while the latter reaches the performance of the former as late as the 19th communication round. This behaviour can be partially explained by Fig. 10, which shows that AdaFed has an overall better F_1-score performance, and by its improved weighted averaging rule, namely its ability to increase the weight of clients that have a more balanced dataset at their disposal.

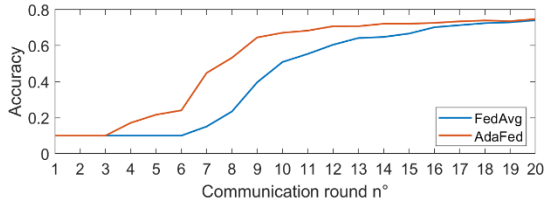

Figure 9. Simulation 2 (CIFAR10): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

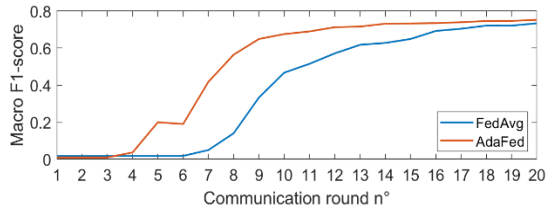

Figure 10. Simulation 2 (CIFAR10): Evolution of the server model macro F_1-score over communication rounds, evaluated on the separated test set of the server.

C. *Simulation 3 - Medical Imaging*

As introduced, for its privacy preserving features, one of the most promising applications of FL is in the healthcare domain [48]. In fact, over the last few years several disruptive ML solutions have been developed to support medical operators and caretakers, but their development and training typically required a complex and expensive data collection campaign subject to several strict regulations such as GDPR.

The FL framework provides an opportunity to enable the collaboration among clinical institutions by abolishing any confidential data exchanges. In this section, we will utilise two different medical datasets to demonstrate some additional characteristics of AdaFed.

1) *Simulation 3.A (NIH malaria)*

In this simulation we test our algorithm on the NIH malaria dataset [49] that consists in 27,558 cell images (of which 10% were reserved for the test set). The task associated to this dataset is a binary classification one and consists in discerning whether the depicted cell is infected. Fig. 11 reports an example of images contained in the dataset.

We considered $K = 5$ clients, figuratively representing five different medical institutions, and we divided uniformly the dataset among them. The neural network we used consisted of a stack of convolutional layers of 32, 32, 64, 64, 128, 128 filters of size 3×3 followed by a fully connected layer of 128 neurons, all with *ReLu* activation functions and a dropout of 0.15. The output layer of the network consisted of a layer with a single sigmoid neuron. The loss considered was, as customary for binary classification tasks, the binary cross-entropy and it was minimized by an ADAM optimizer initialized with the standard Keras parameters. All other parameters were the same as in the first simulation.

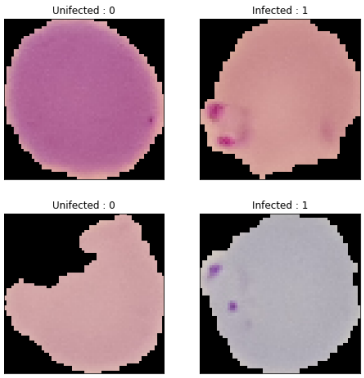

Figure 11. Simulation 3.A (NIH malaria dataset): Example of dataset images.

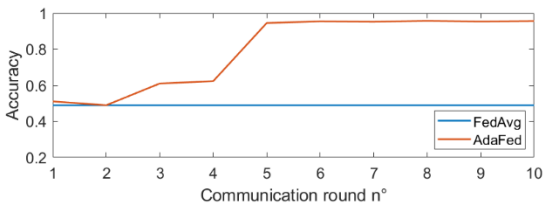

Figure 12. Simulation 3.A (NIH malaria dataset): Evolution of the server model accuracy over communication rounds, evaluated on the separated test set of the server.

The neural network architecture performs very well when trained on the entirety of the dataset, fast reaching a classification accuracy on the test set of over 95% in about three epochs. Conversely, due to the significant level of dropout and the relatively high number of parameters, the same network performed poorly on the individual clients, often becoming insensitive to the input and returning, as prediction, always the (slightly) more common label in the avail- able dataset. This phenomenon compromises the performance of FedAvg, as shown in Fig. 12 (blue line), where the federation does not manage to outperform a random agent. On the contrary, AdaFed is able to discard the models that show this unfavourable behaviour and, by propagating a neural network that combines only the properly trained ones, is able to reach the centralised performance (95%) after about five communication rounds.

As already shown in the test on the OARF classification task (Section IV.*A.4)*), AdaFed demonstrated robustness against adversarial/unfavourable clients, allowing the federation to solve the task at hand.

2) *Simulation 3.B (VGG-Cell) - Regression*

In this final experiment we consider a regression problem, consisting in counting the number of cells that appear in a microscope image. For this task, in [50] the VGG-cell dataset was proposed, consisting of 200 simulated images (e.g., see Fig. 13). The considered federation is formed by $K = 4$ clients (once again representing different medical institutions) with evenly

distributed data. The client's model implemented is the deep autoencoder [51] proposed in [52], that is characterised by over 3M parameters. The autoencoder is to be trained to reconstruct an image that identifies the centers of the cells, so that the image integral (pixel-wise sum) is equal to the cell count. For both AdaFed and FedAvg, for the training of the clients we implemented a data augmentation procedure that rotated and flipped each sample, obtaining eight times the original data. We set $E = 3$, $B = 50$ and $R = 10$. The evaluation of the model performance and loss is carried out on a separated test set consisting in 10% of the original data, which was also augmented as described above.

In this simulation, the model averaging procedure is conducted based on a different performance index with respect to the pixel-wise mean-squared error loss of the models, and it is set as the mean absolute percentage error (MAPE) of the cell counting.

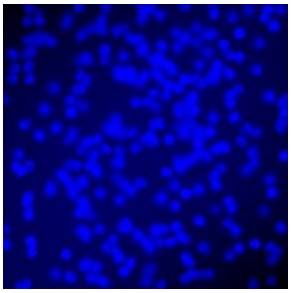

Figure 13. Simulation 3.B (VGG-Cell): Example of dataset image.

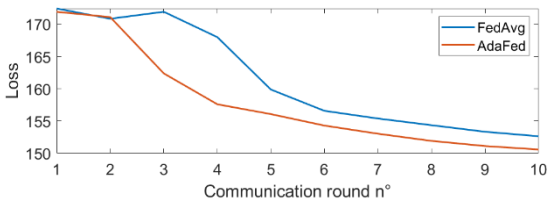

Figure 14. Simulation 3.B (VGG-Cell): Evolution of the server model loss over communication rounds, evaluated on the separated test set of the server.

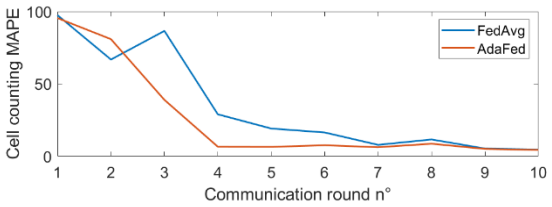

Figure 15. Simulation 3.B (VGG-Cell): Evolution of the server model performance metric (cell counting mean percentage error) over communication rounds, evaluated on the separated test set of the server.

Fig. 14 and Fig. 15 show that also for this regression task AdaFed is able to outperform the FedAvg: the model loss decreases faster and, after 10 rounds, the loss value is 150.6 for AdaFed and 152.6 for FedAvg; the cell count MAPE reaches a value of about 6% at the fourth communication round while FedAvg needs 7 rounds to reach the same value. The main reason behind this behaviour is that AdaFed emphasizes the better performing clients, enabling a more efficient knowledge sharing through the federation. On the contrary, FedAvg gives the same importance to clients that are currently failing the task, consequently affecting and slowing down the convergence of the overall federated model.

V. Conclusion

This paper presented AdaFed, a new Federated Learning algorithm that is based on a weighted model averaging procedure that accounts for the different performance attained by the federation clients.

AdaFed allows to deal with non-IID, imbalanced and extremely distributed data also in the presence of malicious/bad performing federation members. To attain a better performance on complex tasks, the proposed algorithm also envisages the possibility of dynamically adapting the training process itself by employing an adaptive loss function.

Several validation examples were used to show that AdaFed achieves good performance in both classification and regression tasks on challenging scenarios.

Future research directions involve the implementation of privacy-aware features into AdaFed and its decentralization.

Conflict of Interest

The authors declare no conflict of interest.

Author Contributions

The conceptualization of the algorithm and its design were led by A. Giuseppi and A. Pietrabissa. All research activities were conducted under the supervision and coordination of F. D. Priscoli. L. D. Torre and D. Menegatti developed the simulation environment and validated the algorithm. C. Poli was responsible for the identification of the algorithm requirements for healthcare applications and supervised its testing activities. All authors contributed to the writing of the paper and approved its final version.

Acknowledgment

This work has been partially funded by the Lazio region, in the scope of the project FedMedAI, POR FESR Lazio 2014 - 2020 (Azione 1.2.1), Prot. n. A0375-2020-36491 - 23/10/2020.

References

[1] T. Li, A. K. Sahu, A. Talwalkar, *et al.*, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50-60, 2020.

[2] J. Xu, B. S. Glicksberg, C. Su, *et al.*, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, no. 1, pp. 1-19, 2020.

[3] M. J. Sheller, B. Edwards, G. A. Reina, *et al.*, "Federated learning in medicine: Facilitating multi-institutional collaborations without sharing patient data," *Scientific Reports*, vol. 10, no. 1, 2020.

[4] T. Li, A. K. Sahu, M. Zaheer, *et al.*, "Federated optimization in heterogeneous networks," *Proceedings of Machine Learning and Systems*, vol. 2, pp. 429-450, 2020.

[5] Q. Yang, Y. Liu, T. Chen, *et al.*, "Federated machine learning: Concept and applications," *ACM Transactions on Intelligent Systems and Technology*, vol. 10, no. 2, pp. 1-19, 2019.

[6] H. B. McMahan, E. Moore, D. Ramage, *et al.*, "Communication-Efficient learning of deep networks from decentralized data," in *Proc. the 20th International Conference on Artificial Intelligence and Statistics*, 2017.

[7] H. B. McMahan, E. Moore, D. Ramage, *et al.*, "Federated learning of deep networks using model averaging," arXiv preprint arXiv:1602.05629, 2016.

[8] Q. Li, Z. Wen, Z. Wu, *et al.*, "A survey on federated learning systems: Vision, hype and reality for data privacy and protection," *IEEE Transactions on Knowledge and Data Engineering*, 2021.

[9] W. Y. B. Lim, N. C. Luong, D. T. Hoang, *et al.*, "Federated learning in mobile edge networks: A comprehensive survey," *IEEE Communications Surveys & Tutorials*, vol. 22, no. 3, 2020.

[10] K. Bonawitz, V. Ivanov, B. Kreuter, *et al.*, "Practical secure aggregation for privacy-preserving machine learning," in *Proc. ACM SIGSAC Conference on Computer and Communications Security*, 2017.

[11] R. C. Geyer, T. Klein, and M. Nabi, "Differentially private federated learning: A client level perspective," arXiv preprint arXiv:1712.07557, 2017.

[12] M. Nasr, R. Shokri, and A. Houmansadr, "Comprehensive privacy analysis of deep learning: Stand-Alone and federated learning under passive and active white-box inference attacks," in *Proc. IEEE Symposium on Security and Privacy*, 2018.

[13] S. Truex, N. Baracaldo, A. Anwar, *et al.*, "A hybrid approach to privacy-preserving federated learning," in *Proc. the 12th ACM Workshop on Artificial Intelligence and Security*, 2019.

[14] Z. Wang, M. Song, Z. Zhang, *et al.*, "Beyond inferring class representatives: User-Level privacy leakage from federated learning," in *Proc. IEEE Conference on Computer Communications*, 2019.

[15] J. Konečný, H. B. McMahan, F. X. Yu, *et al.*, "Federated learning: Strategies for improving communication efficiency," arXiv preprint arXiv:1610.05492, 2016.

[16] Y. Lin, S. Han, H. Mao, *et al.*, "Deep gradient compression: reducing the communication bandwidth for distributed training," arXiv preprint arXiv:1712.01887, 2017.

[17] F. Sattler, S. Wiedemann, K. R. Muller, *et al.*, "Robust and communication-efficient federated learning from non-i.i.d. data," *IEEE Transactions on Neural Networks and Learning Systems*, pp. 1-14, 2019.

[18] A. Hard, K. Rao, R. Mathews, *et al.*, "Federated learning for mobile keyboard prediction, arXiv preprint arXiv:1811.03604, 2018.

[19] T. Yang, G. Andrew, H. Eichner, *et al.*, "Applied federated learning: Improving google keyboard query suggestions," arXiv preprint arXiv:1812.02903, 2018.

[20] S. Ramaswamy, R. Mathews, K. Rao, *et al.*, "Federated learning for emoji prediction in a mobile keyboard," arXiv preprint arXiv:1906.04329, 2019.

[21] J. Luo, X. Wu, Y. Luo, *et al.*, "Real-World image datasets for federated learning," arXiv preprint arXiv:1910.11089, 2019.

[22] Y. Liu, A. Huang, Y. Luo, *et al.*, "An online visual object detection platform powered by federated learning," in *Proc. the AAAI Conference on Artificial Intelligence*, 2020.

[23] A. Imteaj and M. H. Amini, "Distributed sensing using smart end-user devices: Pathway to federated learning for autonomous IoT," in *Proc. International Conference on Computational Science and Computational Intelligence*, 2019.

[24] T. S. Brisimi, R. Chen, T. Mela, *et al.*, "Federated learning of predictive models from federated electronic health records," *International Journal of Medical Informatics*, vol. 112, pp. 59-67, 2018.

[25] Y. Chen, X. Qin, J. Wang, *et al.*, "FedHealth: A federated transfer learning framework for wearable healthcare," *IEEE Intelligent Systems*, vol. 35, no. 4, 2020.

[26] M. J. Sheller, G. A. Reina, B. Edwards, *et al.*, "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*, Springer International Publishing, 2019, pp. 92-104.

[27] Y. Chen, X. Yang, X. Qin, *et al.*, "Dealing with label quality disparity in federated learning," in *Federated Learning*, Springer, 2020.

[28] J. T. Barron, "A general and adaptive robust loss function," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[29] A. A. Heydari, C. A. Thompson, and A. Mehmood, "SoftAdapt: Techniques for adaptive loss weighting of neural networks with multi-part loss functions," arXiv preprint arXiv:1912.12355, 2019.

[30] C. S. Miranda and F. J. V. Zuben, "Multi-objective optimization for self-adjusting weighted gradient in machine learning tasks," arXiv preprint arXiv:1506.01113, 2015.

[31] B. Teixeira, B. Tamersoy, V. Singh, *et al.*, "Adaloss: Adaptive loss function for landmark localization," arXiv preprint arXiv:1908.01070, 2019.

[32] J. Redmon, S. Divvala, R. Girshick, *et al.*, "You only look once: Unified, real-time object detection," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2016.

[33] Y. Cui, M. Jia, T. Y. Lin, *et al.*, "Class-Balanced loss based on effective number of samples," in *Proc. IEEE/CVF Conference on Computer Vision and Pattern Recognition*, 2019.

[34] T. Y. Lin, P. Goyal, R. Girshick, *et al.*, "Focal loss for dense object detection," in *Proc. IEEE International Conference on Computer Vision*, 2017.

[35] A. Giuseppi, L. D. Torre, D. Menegatti, *et al.*, "AdaFed: Performance-based adaptive federated learning," in *Proc. the 5th International Conference on Advances in Artificial Intelligence*, 2021, pp. 38-43.

[36] Y. LeCun, C. Cortes, and C. J. Burges. (2010). MNIST handwritten digit database. *ATT Labs*. [Online]. Available: http://yann.lecun.com/exdb/mnist/

[37] Official Keras.io documentation: Example of Convolutional Neural Network for MNIST.

[38] S. Hu, Y. Li, X. Liu, *et al.*, "The OARF benchmark suite: Characterization and implications for federated learning systems," *ACM Transactions on Intelligent Systems and Technology*, vol 13, no. 4, 2020.

[39] J. J. McAuley and J. Leskovec, "From amateurs to connoisseurs," in *Proc. the 22nd International Conference on World Wide Web*, 2013.

[40] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1780, 1997.

[41] S. Hu, Y. Li, X. Liu, *et al.*, "Official implementation of the OARF benchmark suite: Characterization and implications for federated learning systems," 2020.

[42] A. Krizhevsky and G. Hinton. (2009). Learning multiple layers of features from tiny images. [Online]. Available: https://www.cs.toronto.edu/~kriz/learning-features-2009-TR.pdf

[43] S. Bozinovski, "Reminder of the first paper on transfer learning in neural networks, 1976," *Informatica*, vol. 44, no. 3, 2020.

[44] S. Bozinovski and A. Fulgosi, "The influence of pattern similarity and transfer learning upon training of a base perceptron b2," *Proceedings of Symposium Informatica*, pp. 121-126, 1976.

[45] J. Deng, W. Dong, R. Socher, *et al.*, "ImageNet: A large-scale hierarchical image database," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, 2009.

[46] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," arXiv preprint arXiv:1409.1556, 2014.

[47] A. F. Agarap, "Deep learning using Rectified Linear Units (ReLU)," arXiv preprint arXiv:1803.08375, 2018.

[48] G. A. Kaissis, M. R. Makowski, D. Rückert, *et al.*, "Secure, privacy-preserving and federated machine learning in medical imaging," *Nature Machine Intelligence*, vol. 2, no. 6, pp. 305-311, 2020.

[49] S. Rajaraman, S. K. Antani, M. Poostchi, *et al.*, "Pre-trained convolutional neural networks as feature extractors toward improved malaria parasite detection in thin blood smear images," *PeerJ*, vol. 6, p. e4568, 2018.

[50] V. Lempitsky and A. Zisserman, "Learning to count objects in images," *Advances in Neural Information Processing Systems*, vol. 23, 2010.

[51] M. A. Kramer, "Nonlinear principal component analysis using autoassociative neural networks," *AIChE Journal*, vol. 37, no. 2, pp. 233-243, 1991.

[52] W. Xie, J. A. Noble, and A. Zisserman, "Microscopy cell counting and detection with fully convolutional regression networks," *Computer Methods in Biomechanics and Biomedical Engineering: Imaging & Visualization*, vol. 6, no. 3, pp. 283-292, 2016.

Alessandro Giuseppi is a Postdoctoral Researcher at the University of Rome "La Sapienza" where he received his master degree in Control Engineering and his Ph.D. degree in Automatica respectively in 2016 and 2019. He is the scientific coordinator of the ARIES project, funded by the ESA in the field of fire management, and since 2016 he has participated in 6 other EU and National research projects. Currently, he is serving as associate editor for the International Journal of Control, Automation and Systems. His main research activities are in the fields of network control and intelligent systems, where he published about 40 papers in international journals and conferences.

Lucrezia Della Torre is a data scientist in the oil industry, and has received from the university of Rome "La Sapienza" her M.Sc. degree in Control Engineering in 2020. Her research interests are related to distributed AI systems.

Danilo Menegatti is a Ph.D. student in Automatica the University of Rome "La Sapienza", where he received his M.Sc. degree in Control Engineering in 2020. His research activities are related to spacecraft control, intelligent systems and their applications for smart healthcare.

Francesco Delli Priscoli was born in Rome, Italy, in 1962. He received the degree in Electronics Engineering (summa cum laude) and the Ph.D. in Systems Engineering from the University of Rome "La Sapienza" in 1986 and 1991, respectively. From 1986 to 1991 he worked in Telespazio. Since 1991, he has been working at the University of Rome "La Sapienza," where, at present, he is Full Professor of "Automatic Control," "Control of Autonomous Multi-Agent Systems," and "Control of Communication and Energy Networks". He is a member of the IFAC Technical Committee on "Networked Systems". He was/is the scientific responsible for 40 projects funded by the European Union and by the European Space Agency (ESA). His research interests concern closed-loop multi-agent learning techniques in advanced communication and energy networks.

Antonio Pietrabissa is Associate Professor at the Department of Computer, Control, and Management Engineering "Antonio Ruberti" (DIAG) of the University of Rome Sapienza, where he received his degree in Electronics Engineering and his Ph.D. degree in Systems Engineering in 2000 and 2004, respectively, and where he teaches Automatic Control and Process Automation. Since 2000, he has participated in about 25 EU and National research projects. Currently, he is the scientific responsible of the European research project 5G-ALLSTARS on 5G communications, funded within the H2020 Europe-South Korea cooperation, the coordinator of the project ARIES on fire emergency prevention, funded by ESA, and the coordinator of FedMedAI project, funded by Regione Lazio. He serves as Associate Editor for Control. Eng. Pract. (Elsevier) and for IEEE Trans. Autom. Sci. Eng. His research focuses on the application of systems and control theory to the analysis and control of networks. He is author of more than 50 journal papers and 80 conference papers.

Cecilia Poli is researcher at the National Center for Innovative Technologies in Public Health (TISP) of the Istituto Superiore di Sanità (ISS). She received her degree in Electronics Engineering and his Ph.D. degree in Applied Physics from University of Rome Sapienza in 2001 and 2006, respectively. Her research interests are in the modelling of bacterial growth curves and in the applications of AI methodologies in the medical field. Her work activity also deals with regulatory requirements of the medical device sector (directives 93/42CEE and 98/79CEE, regulations REG UE 2017/745 and REG UE 2017/746). She was the ISS coordinator for the project DAAS on data analysis in explosion risk areas and is currently the ISS coordinator for the project FedMedAI on medical applications of federated learning, funded by Regione Lazio (IT).