Federated Learning, tests of a new approach to distributed learning on a toy model

Lorenzo Sani

April 22, 2021

Contents

1 Abstract

The objective of this short report is to present the main properties of the Federated Learning approach by testing it on a simple toy model. Then, by modifying slightly the basic dataset, some more features are exploited for future use on real problem.

2 Introduction

Federated Learning (FL) is a newly introduced approach to collaborative machine learning [1]. There are nowadays many traditional machine learning algorithms which require huge quantities of data raining examples to learn. The rising problem is that is often very difficult to collect a sufficient amount of data to reach a reasonable reliability. This problem is very common in settings in which data access is restricted by righteous privacy regulations, e.g. personal healthcare, or customization of personal devices. The implementation of a FL setting allows to avoid the necessity to collect data in a single place.

2.1 Federated Learning

A typical FL setting is built by taking into account a set of clients and a centralized server. The main steps of a FL iterative algorithm are the following:

1. global model initialization;
2. global model distribution;
3. local models training;
4. local models aggregation;
5. repeat from 2.

Firstly, step (1), the server, usually, initialize a unique global model to be passed to the clients and keep the consistency of the procedure, i.e. ensuring that every client runs the same learning algorithm. Then, step (2), the server is in charge to distribute the initialized model to all the available clients. FL algorithms allow to check how many clients are available and train the same even if they are a low number. The distribution procedure could take into account the numerosity of the single clients' datasets (the only one information on the dataset coming from the clients). After having received the global model, all the available clients train locally the global model with their data, step (3), producing a new local model. The weights of all the local models are then aggregated, step (4), at the server place to be reduced to a new global model. The aggregation procedure, as the distribution previously, could take into account the numerosity of the clients' datasets. Moreover the evaluation step, always performed locally by the clients, could be aggregated to the server in an analogous procedure than local models. This allows the server to have global perception of the performance of the learning step by step. This action usually is inserted between step (2) and step (3). The iterative procedure is finally repeated until the number of fixed federated iterations is reached, usually.

2.2 Federated Average (FedAvg)

Before presenting the methods used, it is necessary to spend some words on the aggregation method used. Despite FL is a newly introduced approach, many aggregation procedures have been proposed. It is important to say that FL is not widely understood by now and because of this not any aggregation procedure, or more generally FL algorithm, is reliable for any specific problem. The aggregation method used in this analysis is, probably, the most general: `FedAvg`. This algorithm, firstly proposed in [2], relies on Stochastic Gradient Descent (SGD) optimization method, since the majority of the most successful deep learning works were based on this. The available clients locally compute (step 3) their average gradient on their local data at the current model w_t, where t identifies the federated round, and the central server aggregates these gradients and applies the update $w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} g_k$. Above, $g_k = \nabla F_k(w_t)$ is the average gradient of the client k, η is the learning rate, n_k is the number of samples at the client k, n is the total number of samples (sum over all the available clients). Equivalently, the update can be given by $w_{t+1} \leftarrow w_t - \eta \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$, where $w_{t+1}^k \leftarrow w_t - \eta g_k \; \forall k$. In the last, every client takes a complete step of gradient descent, while the server only takes the weighted average of the resulting models. The following pseudo-code summarizes the procedure.

Algorithm 1: *FedAvg*. The K clients are indexed by k; E is the number of local epochs; η is the learning rate; B is the size of the local batches.

```
Server executes:
    initialize $w_0$
    for each federated round $t = 1, 2, ...$ do
        $S_t \leftarrow$ (select available clients)
        for each client $k$ in $S_t$ do in parallel
            $w_{t+1}^k \leftarrow$ ClientUpdate($k$, $w_t$)
        end
        $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$
    end
ClientUpdate($k$, $w_t$):
    $\mathcal{B} \leftarrow$(split $k$-th client's dataset into batches of size $B$)
    for each local epoch $i$ from 1 to $E$ do
        for batch $b \in \mathcal{B}$ do
            $w \leftarrow w - \eta \nabla l(w; b)$
        end
    end
    return $w$ to server
```

3 Methods

3.1 Dataset

A simple toy dataset was chosen to set up a classification toy model to perform some simulations in FL setting. From the `scikit-learn` Python package, which provides a wide set of generators for toy datsets, the `datasets.make_moons` generator was picked up. This function produces the requested number of points in a 2-D space drawing two interleaving circles, as Fig.1 shows. The same function

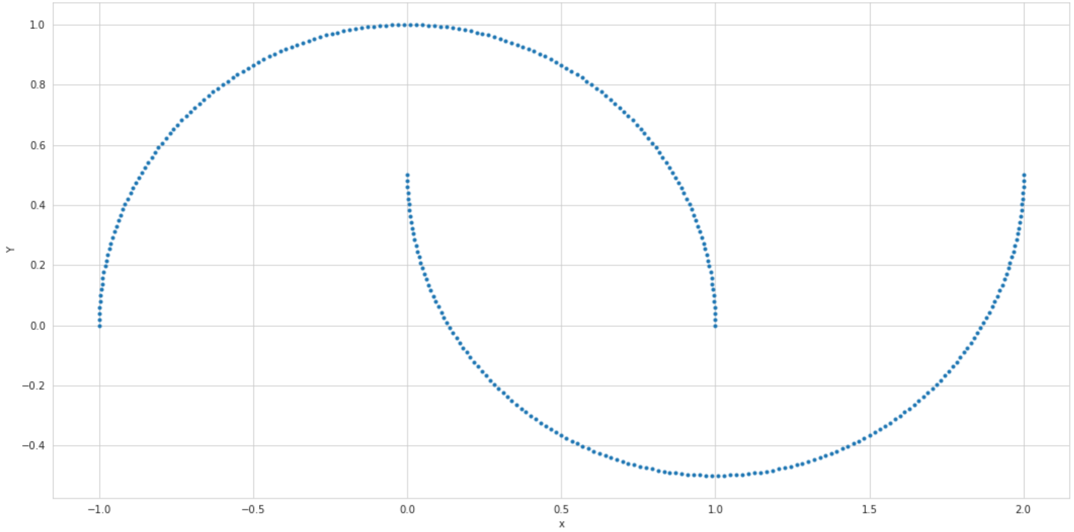


Figure 1: Here are represent an example of a set of 320 points of the toy dataset, i.e. two interleaving circles. These points are taken by setting noise to zero

returns also the classification array, that relates every point to its corresponding circle. One can make the request to add some noise to the generated points, and ask for the points to be shuffled, once generated. The noise value was fixed to 0.1 along every simulation. It is also possible to set the random state that seeds the noise and shuffling, if requested. Two more functions, in addition to this settings, were built to transform a little such generated dataset. The first simply translates the dataset by a given vector (dx, dy), i.e. every point (x, y) in the dataset undergoes the transformation $x' = x + dx$ and $y' = y + dy$. The second applies a simple rotation by an angle θ with a standard transformation, i.e. $x' = x\cos(\theta) - y\sin(\theta)$ and $y' = x\sin(\theta) + y\cos(\theta)$ following the above notation. Examples of translated and rotated datasets are shown in Fig.2.

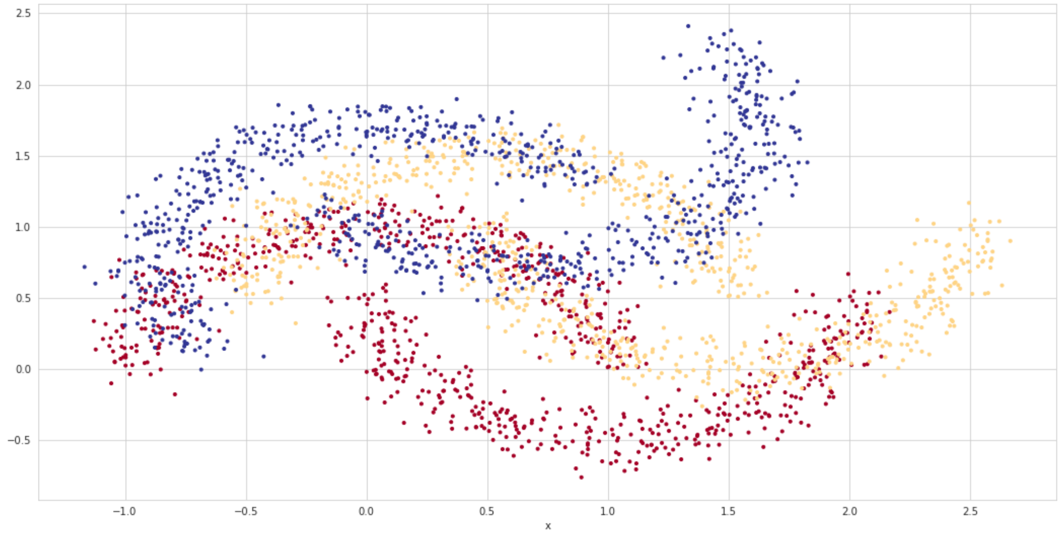


Figure 2: Here are shown three examples of the toy datasets. Are taken 840 points for every dataset, with noise fixed at 0.1. The red dots represent the standard toy dataset. The blue dots represent the rotated dataset by an angle of $\theta = \pi/5$. The yellow dots represent the translated dataset using $dx = 0.5$ and $dy = 0.5$

3.2 Model

Every simulation was build on a simple two layers sequential model. Both the layers are standard regular densely-connected Neural Network layers, the first with 4 output, the second with 2, since the model is expected to classify the points w.r.t. their circle of belonging.

3.3 FL framework

The framework adopted to make the necessary simulation is Flower: A Friendly Federated Learning Framework [3]. The federated framework is set up by running a program for every client and one for the server. These will communicate using a RPC (Remote Procedure Call) framework exchanging the weights of the model. The ML framework used is `tensorflow` with a `keras.Sequential` model. The client program accepts many parameters to build properly the client's dataset, and to set up the outputs also. The server program has a minimal configuration, receiving the number of clients in the federated network and the number of the federated epochs to run. Another simple program runs an aggregated model by building an equivalent aggregated dataset. Every client's dataset, as the aggregated one, is divided randomly in train set and test set using the standard proportion 80-20, for train and test respectively. The for every epoch an evaluation step is performed at every client's place as at the aggregated model. The performance of the model at every step, represented by the loss and the accuracy, is retrieved and saved to be consulted later. The loss function chosen is the Sparse Categorical Cross-Entropy, implemented by the function `tensorflow.keras.losses.SparseCategoricalCrossentropy`. The accuracy is simply computed as the ratio between the number of well classified points and the total number of points in the test set.

3.4 Simulations details

In the very first simulation, different FL set ups are compared with the aggregated one. The number of epochs (federated epochs for FL set ups, "standard" epochs for the aggregated model) is fixed to 1000 for every set up, in order to compare easily the performances against the epochs. The number of total samples is fixed to 840 points, which are equally distributed between the chosen number of clients. A number of 6 different FL settings are chosen, with 2,3,4,5,6,7,8 clients respectively. The performance, keeping fixed the number of epochs, is expected to be worst as the number of clients increases.

The subsequent simulation is intended to compare the performance of 6 different FL set ups, with 2,3,4,5,6,7,8 clients respectively, between each other. This time the number of federated epochs is again fixed to 1000, but the number of local epochs, performed at clients' places, are chosen to increase as the number of clients does, i.e. 1,2,2,3,3,4,4 local epochs respectively. The number of total samples is again fixed to 840 points, which are distributed as before. This choice allows the FL models to reach faster their best performances as higher is the number of clients in FL set up. The final result should show how much the best performance of a FL set up depends on the number of clients chosen to distribute the dataset.

The last simulation attempts to better understand the Transfer Learning (TL) ability of such a FL setting. In order to do this, different 8 clients FL set ups are compared. This time different FL set ups carry very different clients, in fact a portion of the clients is transformed using the translation transformation defined above with $(dx, dy) = (0.2, 0.2)$. There were taken 8 simulation with the 8 different proportion between translated and standard dataset, from 1/8 to 8/8. An equivalent simulation was done using the rotation transformation with $\theta = \pi/5$. The wideness of the confidence interval in the mean losses and mean accuracies plots of these FL set ups should give information on the TL ability, at least in comparison w.r.t. the other FL set ups.

4 Results

The easiest method to compare the performances of such models is to plot the losses and the accuracies against the number of epochs (federated epochs for FL set ups, "standard" epochs for the aggregated model). Moreover, to have a general, and more informative, idea of the performance of the whole FL set ups, the mean values of the losses and the accuracies of all the clients in every FL set up are computed. A confidence interval, that covers a confidence level of 68%, i.e. standard deviation was taken, is then associated to these mean values, in order to give a more complete representation.

The very first result to notice is that the aggregated model has a definitely faster convergence, for both the loss and the accuracy, w.r.t. any other FL set up. More the FL set up is distributed, slower is the convergence, as Fig.3-4 show, and higher is the standard deviation between the clients.

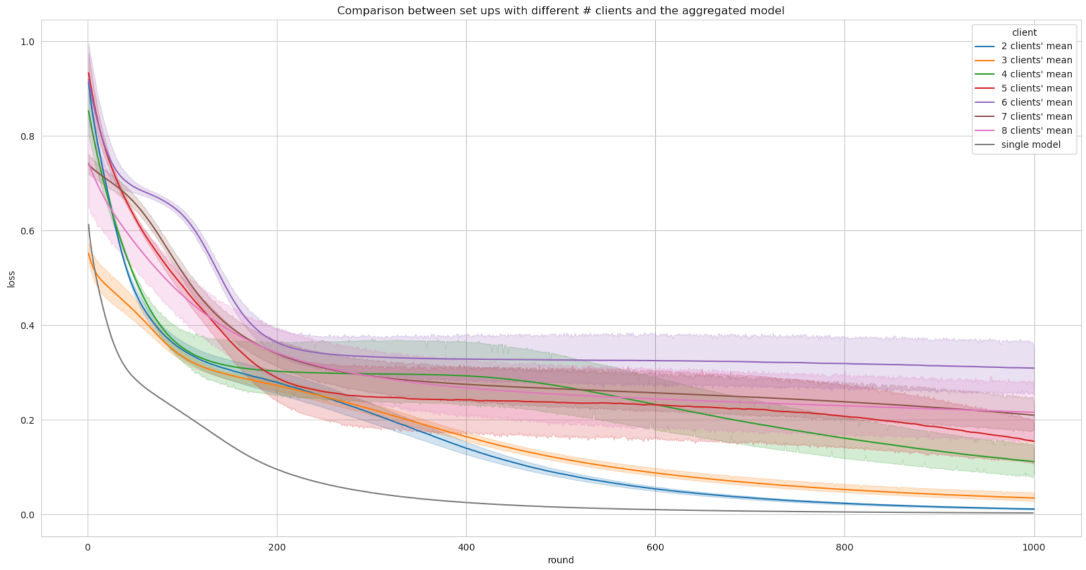


Figure 3: Here is represented the evolution of the loss function for different set ups, summarized in the legend, in the number of epochs. For every curve, except for the one referring to the aggregated model, is provided a confidence interval covering the 95% of the true value.

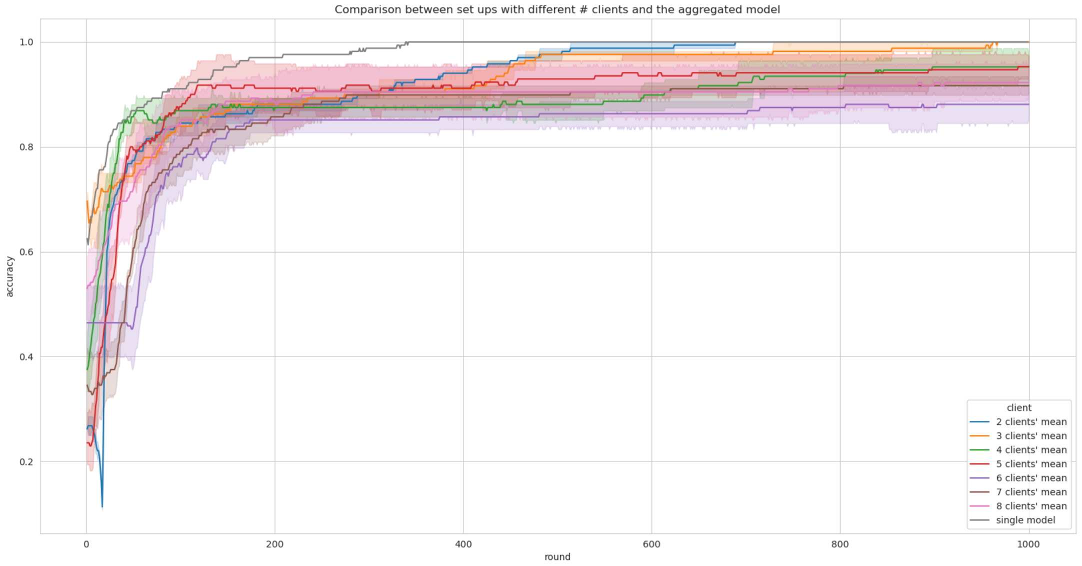
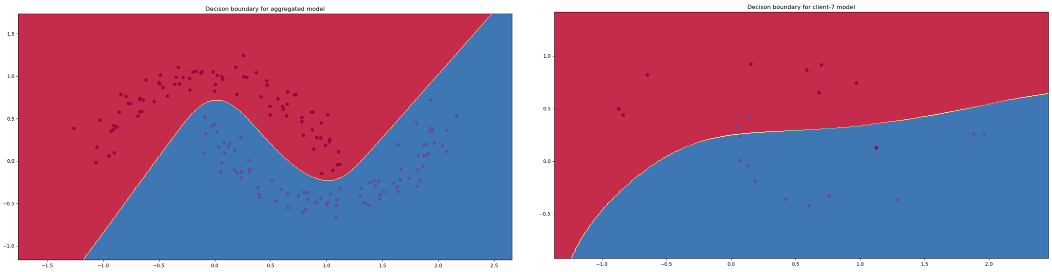


Figure 4: Here is represented the evolution of the accuracy function for different set ups, summarized in the legend, in the number of epochs. For every curve, except for the one referring to the aggregated model, is provided a confidence interval covering the 95% of the true value.

The standard deviations are expected to tend to zero as the number of epochs increases. A useful comparison is shown by Fig.5, underlining, using the final decision boundaries, the fact that with only 1000 federated epochs, the FL setting distributed between 8 clients is far from the aggregated one.



(a)

(b)

Figure 5: Here is provided a comparison between the final decision boundaries plotted on the test dataset relative to (a) the aggregated model and (b) the $8-th$ client of the FL set up.

As Fig.6-7 show, the general setting changes, in fact increasing the number of local epochs at clients' place make the learning curves faster than before. One can notice that somewhere in the plot, FL settings with an higher number of clients outperforms others with a lower number of clients.

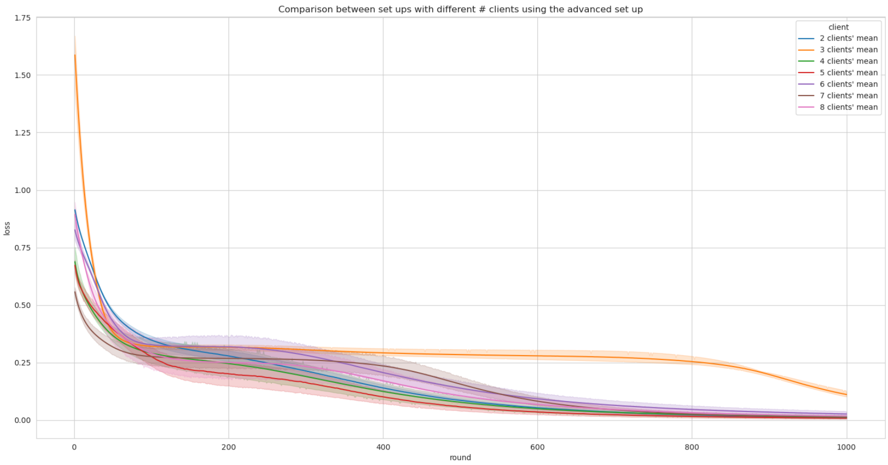


Figure 6: Here is represented the evolution of the loss function for different set ups, summarized in the legend, in the number of epochs. For every curve, is provided a confidence interval covering the 95% of the true value.

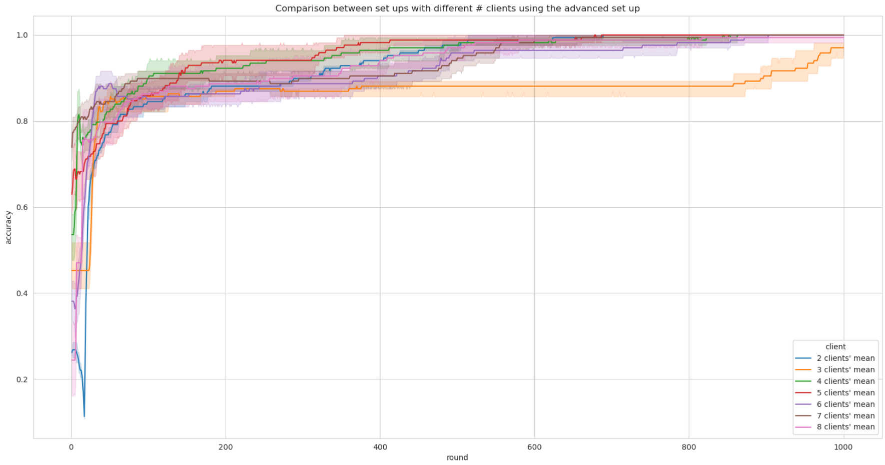


Figure 7: Here is represented the evolution of the accuracy function for different set ups, summarized in the legend, in the number of epochs. For every curve, is provided a confidence interval covering the 95% of the true value.

The last analysis, which tries to seek the TL ability of FL setting, shows that the set ups which perform better are those which are more mixed up. The simulation that used the rotation transformation is almost inconclusive, since the convergence is not reach in many set ups, as shown by Fig.8-9. However Fig.9 shows the general behavior mentioned previously.

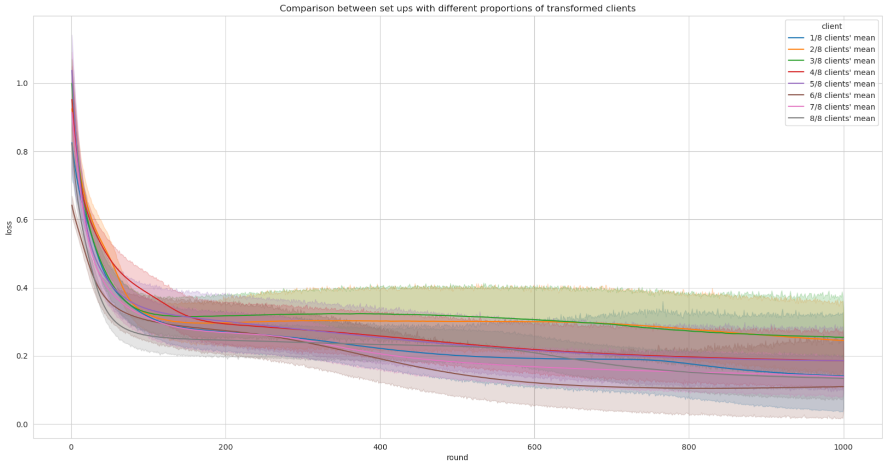


Figure 8: Here is represented the evolution of the loss function for different set ups, summarized in the legend, in the number of epochs. For every curve, is provided a confidence interval covering the 95% of the true value. The fractions that define the set ups in the legend refer to the fraction of clients in the set up having a rotated dataset.

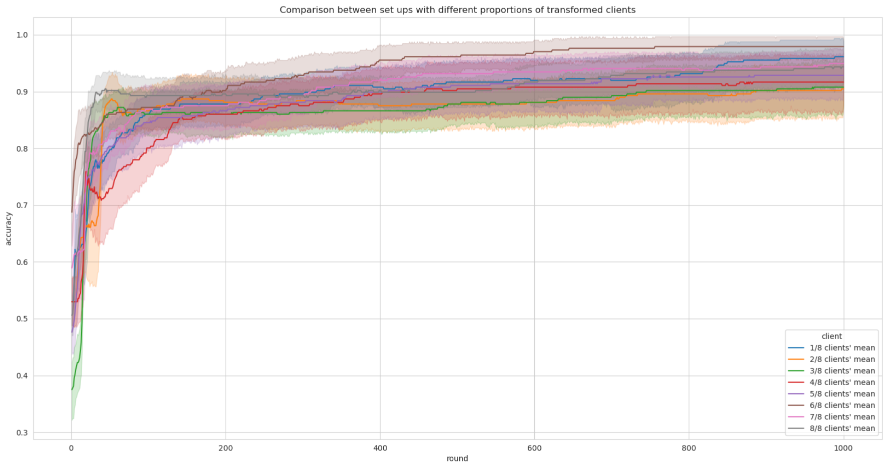


Figure 9: Here is represented the evolution of the accuracy function for different set ups, summarized in the legend, in the number of epochs. For every curve, is provided a confidence interval covering the 95% of the true value. The fractions that define the set ups in the legend refer to the fraction of clients in the set up having a rotated dataset.

The simulation that used the translation transformation has many set ups, which nearly reach the convergence, describe better the general behavior noticed. As Fig.10-11 show, the worst performances are those of the FL settings with 7 and 8 clients with transformed datasets, then set ups with 1 and 2 clients with transformed datasets follow.

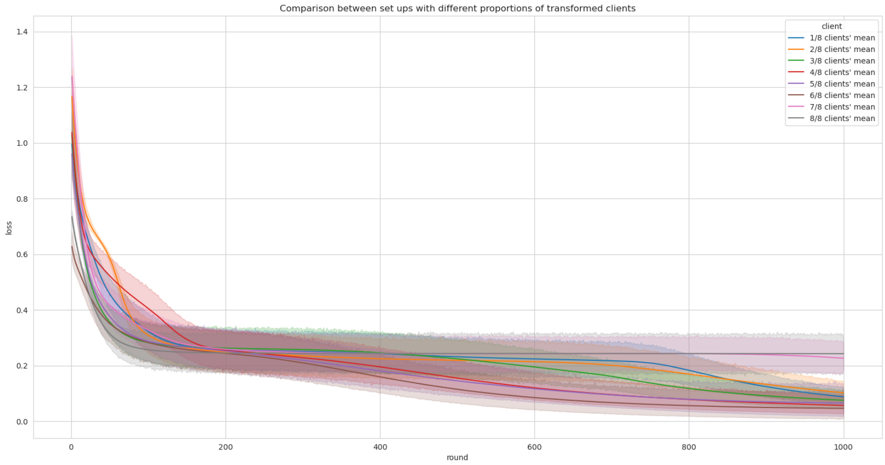


Figure 10: Here is represented the evolution of the loss function for different set ups, summarized in the legend, in the number of epochs. For every curve, is provided a confidence interval covering the 95% of the true value. The fractions that define the set ups in the legend refer to the fraction of clients in the set up having a translated dataset.

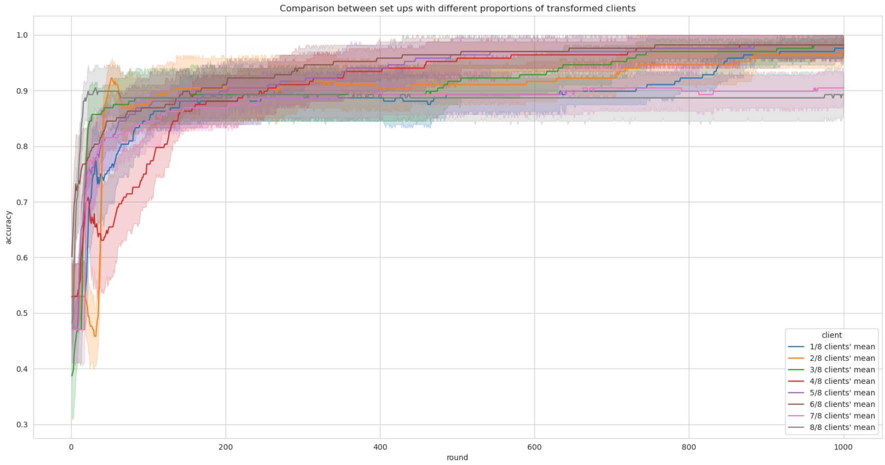


Figure 11: Here is represented the evolution of the accuracy function for different set ups, summarized in the legend, in the number of epochs. For every curve, is provided a confidence interval covering the 95% of the true value. The fractions that define the set ups in the legend refer to the fraction of clients in the set up having a translated dataset.

References

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

[2] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data, 2017.

[3] Daniel J. Beutel, Taner Topal, Akhil Mathur, Xinchi Qiu, Titouan Parcollet, Pedro P. B. de Gusmão, and Nicholas D. Lane. Flower: A friendly federated learning research framework, 2021.