Hybrid Approaches to Federated Learning for Heterogeneous Data Distributions

Erfan Bayat
s328390
Politecnico di Torino
s328390@studenti.polito.it

Elia Faure Rolland
s324224
Politecnico di Torino
s324224@studenti.polito.it

Valeria Petrianni
s331722
Politecnico di Torino
s331722@studenti.polito.it

Abstract

Federated Learning (FL) has emerged as a promising approach for training machine learning models across decentralized data sources without sharing local datasets, thereby addressing privacy concerns and increasing scalability. Our project explores the application of FL in the domains of image classification and language models, comparing it with traditional centralized training methods and simulating the main challenges of FL such as non-IID (non-Independent and Identically Distributed) data distribution and selective client participation. To address the problems often observed in federated settings, especially under non-IID conditions, we propose a pre-trained model on a different dataset as a first option, and an innovative approach that integrates a client's model personalization method (FedRep) with the widely-used FedAvg algorithm as a second option. Our hybrid formulation aims to leverage the robustness of FedRep in handling personalized representations while maintaining the convergence efficiency on a global scale. We conduct experiments and compare the results with the standard algorithms, demonstrating the improvements and trade-offs achieved with our new techniques. The implementation of our code is reported in here.

1. Introduction

Federated Learning is a novel machine learning approach specifically designed for distributed systems, where multiple clients collaboratively train a model while ensuring that their data remains decentralized.
In this architecture, each client performs a predefined number of local training steps (gradient updates) on its dataset and sends the results to a central system. The global parameters are then learned by averaging the updates from all clients on the server side. The main idea behind federated learning is thus to maintain data locality, without sharing local data with the server or among different clients.
The standard aggregation function for federated learning is currently FedAvg [7], which consists of computing a weighted mean among all the parameters of the local results. While federated learning is extremely useful for mitigating privacy concerns [4] and reducing the need for significant computational power, it also introduces new challenges.
Client availability and latency can significantly affect the performance of the federated algorithm. Additionally, in real-world scenarios, clients often have very different data distributions, violating the main IID (Independent and Identically Distributed) assumption in machine learning. This heterogeneity makes learning a robust global model particularly challenging.

1.1. Building a centralized baseline

As a first stage of our work, in order to be able to compare the results obtained with the federated learning approach, we build two centralized models respectively for CIFAR-100 and Shakespeare datasets.
In particular, we train a Convolutional Neural Network (CNN) for the image classification task and a LSTM model for the next character prediction, applying hyperparameters tuning. These neural architectures are the same that will be used for the federated approach.

1.2. Building a federated baseline

In the federated learning scenario, the training phase occurs on the client-side, where each participant trains its own model using its respective dataset. The server node does not directly train the model itself but instead applies validation on a global test set.
The algorithm operates over a total of R global rounds managed by the server node. In each round, the current model is distributed to all clients, which perform local updates and return their updated models to the server. The server then aggregates these values to update the global model.
For this phase we use the FedAvg approach [7], proposed by McMahan et al.
To establish a federated learning baseline, we train a model

using the *default* settings, assuming the IID hypothesis and uniform clients selection.

1.3. Simulating FL drawbacks

The two primary new challenges introduced by FL are:

1. *Client availability.* In real-world scenarios, various clients may possess differing computational resources. Communication with the server node may be impacted by varying latencies, and clients may not always be consistently available. To simulate these aspects, at each round we select a portion of clients among all and we train the local models with respect to only those nodes.
   Our first selection is based on a random extraction. Since some devices may be more available than others, we then try to select clients using the Dirichlet function, which simulates the different probability for a participant to be chosen.

2. *Non-IID distribution of clients' data.* In a more realistic approach, local datasets can be vary significantly from one another. Clients may have a really different distribution of labels, and they can even have different classes among them. This can severely interfere with the main assumption of standard distribuited learning where all data are assumed to be independent and identically distributed.
   We mimic this behaviour by applying a sharding function that splits the full global dataset in local datasets: this function assigns to each client an approximately equal amount of samples, belonging to a specified number N_C of distinct classes.

1.4. Addressing FL challenges with hybrid methods

The aspects described in 1.3 can severely reduce federated learning performances. In particular, as we show in Section 4, the non-IID distribution of clients data has a huge impact on the results in terms of accuracy, especially with smaller values of N_C. The main explanation of this phenomenon is that it is intensely difficult for the global model to approximate the distribution of all clients, if they are markedly different.
This problem can be partially mitigated by adopting the approach proposed by Collins et al. [3], FedRep, which involves creating a personalized version of the global model for each local entity.
In essence, FedRep focuses on building a global model with shared parameters (body) across all participants, while allowing customization of the last layers (head) of the global model for each client locally.
This strategy turns out to be very beneficial in creating tailored local models, while continuing improving the general one.
Nevertheless, each client updates only one time the shared parameters and this may cause a too slow improvement of the global body.
To tackle this problem, we initially attempt to apply FedRep to a pre-trained model on ImageNet. However, in the federated context, a pre-trained model or a centralized dataset might not always be available, making this strategy not suitable for all situations.
As a solution, we propose a variation of FedRep called FedAvg2Rep, in which we run a predefined number of warm-up rounds of the standard FedAvg algorithm described in 1.2 and we then start applying FedRep on the pre-trained global model. By doing this, the central structure is able to learn the shared body during the warm-up phase, and we can create individual clients' models with a robust backbone.

2. Methods

In this section, we formally describe the FedAvg method, which we use for implementing the FL baseline, and we show the full algorithm that we implement for this phase. Subsequently, we describe the FedRep algorithm and our proposal in detail.

2.1. FedAvg

FedAvg consists of performing a number R of communication rounds. Each round can be split into two parts:

1. *Client Training*: in this stage, each client initializes its local model as the model received as input, and performs a number of stochastic-gradient updates equal to J.

2. *Server Update*: after all the local training are finished (or at least a subset of them), the aggregation of all the local results is computed by the server node.

At a generic time t, given a global model w_{ϕ^t}, and K local models $w_{\phi_k^t}$ from clients with local datasets of dimensions n_k, the weighted FedAvg aggregation function can be represented as follows:

$$w_{\phi^t} = \sum_{k=1}^{K} \frac{n_k}{N} w_{\phi_k^t} \tag{1}$$

If we assume that each client has approximately the same amount of data, we assign to each of them the same relevance, reshaping equation 1 as:

$$w_{\phi^t} = \frac{1}{K} \sum_{k=1}^{K} w_{\phi_k^t} \tag{2}$$

The full algorithm is represented in Algorithm 1.

Algorithm 1 Federated Averaging (FedAvg) Algorithm

```
Require: $K$ clients, number of communication rounds $R$
1: Initialize global model parameters $w_{\phi^0}$
2: for each round $t = 1$ to $R$ do
3:     for each client $k = 1$ to $K$ in parallel do
4:         // Client $k$ performs local computation:
5:         $w_{\phi_k^t} \leftarrow \text{ClientUpdate}(w_{\phi^{t-1}}, k, J)$
6:     end for
7:     Aggregate local model updates at the server:
8:         $w_{\phi^t} = \frac{1}{K}\sum_{k=1}^{K} w_{\phi_k^t}$,
9: end for
10: return Updated global model parameters $w_{\phi^R}$

1: Function ClientUpdate($w_{\phi^{t-1}}$, $k$, $J$)
2:     // Initialize local model parameters $w_{\phi_k^{t,0}} \leftarrow w_{\phi^{t-1}}$
3: for $j = 1$ to $J$ do
4:         // Perform local model update:
5:         Train local model on client $k$'s dataset for one iteration
6:         Update local model parameters $w_{\phi_k^{t,j}}$
7: end for
8: return Updated local model parameters $w_{\phi_k^{t,J}}$
```

2.2. FedRep

The FedRep approach, contrarily to FedAvg, in which the entire model is trained locally for a number of steps J, consists of a different version of the ClientUpdate function shown in 1.
At each communication round, each client receives the global model ϕ^t and updates the weights of the head $w_{h^{t,j}}$ of the shared model with respect to the sample $\mathcal{P}_j$ at that step, for a number of steps $J-1$, in the following way:

$$w_{h^{t,j}} := w_{h^{t,j-1}} - \eta \frac{1}{|\mathcal{P}_j|} \sum_{i \in \mathcal{P}_j} \frac{\partial f_i}{\partial w}(w_{h^{t,j-1}}) \qquad (3)$$

where f is the objective loss function and η represents the step size. After this passage, it computes one single gradient update of the weight of the entire model $w_{\phi^{t,J-1}}$, as follows:

$$w_{\phi^{t,J}} := w_{\phi^{t,J-1}} - \eta \frac{1}{|\mathcal{P}_J|} \sum_{i \in \mathcal{P}_J} \frac{\partial f_i}{\partial w}(w_{\phi^{t,J-1}}) \qquad (4)$$

The final result is sent back to the server, which aggregates all the results. The ClientUpdate function suggested by FedRep is represent in Algorithm 2.

2.3. Global model warm-up

It follows from function 2 that, for each client, the last layer of the model is updated $J-1$ times, while the backbone of the model is update just one time at the end.
Even though if this very last passage is crucial for the improvement of the model on a global scale, the body may require a large number of communication rounds before reaching a good state. This behaviour is shown in section 4.
To address this issue, we propose warming up the global model before applying FedRep. This approach aims to establish a partially optimized backbone for the global model, allowing us to focus on client personalization.
We propose two primary solutions to achieve this:

- *Pretraining on a different dataset.* We proprose centrally pretraining the global model for a number of epochs E on Imagenet before running FedRep. This aligns with algorithm 1, where the input model is a pre-trained one.

Algorithm 2 FedRepClientUpdate Function

```
1: Function FedRepClientUpdate($w^{(t-1)}$, $k$, $J$)
2:     // Initialize local model parameters $w_k^{(t-1)} \leftarrow w^{(t-1)}$

3: for $j = 1$ to $J-1$ do
4:         // Perform local model head update:
5:         $w_{h^{t,j}} := w_{h^{t,j-1}} - \eta \frac{1}{|\mathcal{P}_j|} \sum_{i \in \mathcal{P}_j} \frac{\partial f_i}{\partial w}(w_{h^{t,j-1}})$
6: end for
7:     // Perform full local model update:
8:     $w_{\phi^{t,J}} := w_{\phi^{t,J-1}} - \eta \frac{1}{|\mathcal{P}_J|} \sum_{i \in \mathcal{P}_J} \frac{\partial f_i}{\partial w}(w_{\phi^{t,J-1}})$
9: return Updated local model parameters $w_{\phi^{t,J}}$
```

Algorithm 3 FedAvg2Rep

```
Require: $K$ clients, global model $w$, $T_1$, $T_2$
1: Initialize global model parameters $w^{(0)}$
2:
3: //Apply FedAvg for $T_1$ rounds
4: for each round $t = 1$ to $T_1$ do
5:     for each client $k = 1$ to $K$ in parallel do
6:         $w_k^{(t)} \leftarrow \text{ClientUpdate}(w^{(t-1)}, k, \text{local\_step})$
7:     end for
8:     Aggregate local model updates at the server:
9:         $w^{(t+1)} = \frac{1}{K}\sum_{k=1}^{K} w_k^{(t)}$
10: end for
11:
12: //Apply FedRep for $T_2$ rounds
13: for each round $t = T_1 + 1$ to $T_1 + T_2$ do
14:     for each client $k = 1$ to $K$ in parallel do
15:         $w_k^{(t)} \leftarrow \text{FedRepClientUpdate}(w^{(t-1)}, k, \text{local\_step})$
16:     end for
17:     Aggregate local model updates at the server:
18:         $w^{(t+1)} = \frac{1}{K}\sum_{k=1}^{K} w_k^{(t)}$
19: end for
20: return Updated global model parameters $w^{(T_1+T_2)}$
```

- *Warm-up with FedAvg.* We apply a number of federated rounds R_1 using FedAvg algorithm described in 2.1, we then start applying Algorithm 2 for a number of rounds R_2. FedAvg2Rep, detailed in 3, aims to advance FedRep in scenarios where a pre-trained model is unavailable and the previous approach cannot be applied.

3. Experimental setup

This section details the framework in which our experiments are conducted.

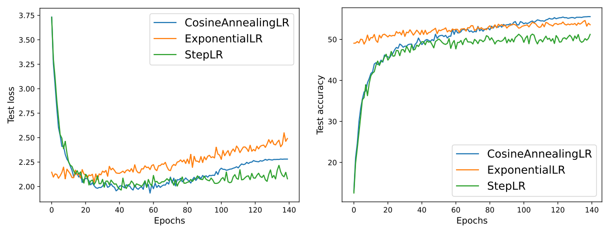
3.1. Datasets

For our experiments we mainly use two datasets: CIFAR-100 and Shakespeare.
CIFAR-100 [9] is a widely-used dataset consisting of 60,000 32x32 color images categorized into 100 classes, with each class containing 500 training images and 100 testing images. It serves as a benchmark for evaluating image classification models.
The Shakespeare dataset, sourced from Leaf [2], is designed for next-character prediction tasks and is specifically structured for federated learning experiments, with data already partitioned among clients. This dataset is available in both IID and non-IID configurations. In the non-IID setup, each speaking role within each play is treated as an individual client.
For our experiments, we select 100 clients from the original 1129, assigning 2000 records per client using an approach inspired by [1]. To establish the centralized baseline, we aggregate data from all participants into a unified dataset used for training and validation, with a separate test set provided.
Tiny ImageNet [6] is used to pretrain the global model before applying FedRep. It comprises 100,000 64x64 colored images across 200 classes. Each class includes 500 training images, 50 validation images, and 50 test images, downsized to 32x32 pixels to align with our CNN architecture.

3.2. Network architectures

3.2.1 FakeLeNet-5

For CIFAR-100, the model architecture is a convolutional neural network similar to LeNet-5 as used in [5], renamed FakeLeNet-5. It has two 5×5, 64-channel convolution layers, each precedes a 2×2 max-pooling layer, followed by two fully-connected layers with 384 and 192 channels respectively and finally a softmax linear classifier. We conduct hyperparameter tuning on the learning rates and batch sizes to identify the optimal configuration. The tested values are detailed in Table 1.



(a) Test loss (b) Test accuracy

Figure 1. Test loss and test accuracy obtained from training FakeLeNet-5 on CIFAR-100 using different learning schedulers.

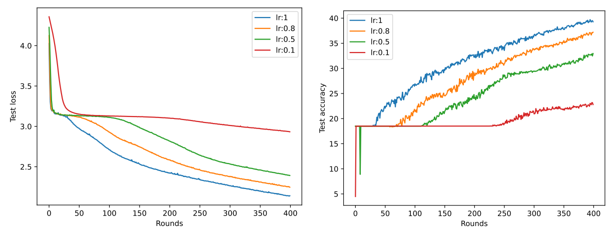
Hyperparameter	**Values**
Learning rate	$\{0.1, 0.01, 0.001\}$
Batch size	$\{32, 64\}$

Table 1. Hyperparameters considered for FakeLeNet-5 and CharLSTM in the centralized setting.

In the centralized context, the optimal configuration includes a learning rate of 0.01 and a batch size of 64. Additionally, we set the weight decay to 4×10^{-4} and the momentum to 0.9; the Cosine Annealing scheduler is employed (the performance with multiple schedulers was tested, as reported in Figure 1).
In the FL experiments, the learning rate is increased to 0.1, the momentum is not applied and the weight decay is set to 4×10^{-3}.

3.2.2 CharLSTM

For the next-character prediction task, applied to Shakespeare dataset, we employ a two-layer Long Short-Term Memory (LSTM) network with an embedding size of 8 and a hidden size of 256, as proposed in [8]. A standard grid-search with the same values used for FakeLeNet-5 (Table 1) is applied to identify the optimal hyperparameter configuration.
The learning rate and batch size are set to 0.01 and 64 respectively in the centralized setting and the Cosine Annealing scheduler is used. We use a momentum of 0.9 and a weight decay of 4×10^{-4}.
Concerning the FL setup, Shakespeare dataset turns out to be affected by a point of local minimum of the loss function, which is well represented in Figure 2a, and creates some difficulties in improving the model's accuracy over 18.5%. This problem can be partially solved by increasing the number of local steps for each client, or by increasing the learning rate used during the training phase.
We further fine-tune the learning rate and determine a new value of 1 for this task. A significantly high learning rate can accelerate escaping local minima in such tasks. This obtained result aligns with observations from [1].



(a) Loss (b) Test accuracy

Figure 2. Training on Shakespeare with different learning rates in the federated setting.

4. Results

In this section, we report and comment the performance of our proposed federated learning approaches on the CIFAR-100 image classification dataset and the Shakespeare language model dataset.

4.1. Centralized Baseline on CIFAR-100

To build our centralized baseline from the CIFAR-100 dataset, we first extract 10,000 records from the training data to serve as the validation split.
Standard data augmentation techniques are applied, and the model is trained for 140 epochs. The training and test accuracy curves demonstrate a steady improvement, achieving a final test accuracy of 55.6%. This baseline serves as a reference point for evaluating the performance of our federated learning approach.

4.2. Centralized Baseline on Shakespeare

As mentioned in Section 3.1, we implement the centralized model for the Shakespeare dataset by merging data from all clients into a single dataset. This merged dataset is subsequently split to obtain a validation set, which is used for hyperparameter tuning. Each sequence provided by the Leaf dataset contains 80 characters, corresponding to the number of distinct possible characters. These values represent the input and output sizes of our model, respectively.
Training the model with the optimal configuration (as reported in Section 3.2.2) yields an accuracy of 52.1% on the test set.

4.3. The first FL baseline

In order to establish a benchmark for our main experiments, we create a first FL baseline for both CIFAR-100 and Shakespeare. In each of the two cases, the key parameters are set as follows: the number of clients K is fixed at 100, and the fraction of clients C participating in each communication round is set to 0.1. The training set is partitioned using an independent and identically distributed sharding method to ensure a balanced distribution of data across clients. In particular, for CIFAR-100 the training dataset is shuffled and split into 100 subsets, each containing the same amount of samples equal to 500.
Concerning Shakespeare, the Leaf repository already provides an IID split of the data [2]. Each client performes $J = 4$ local training steps using their respective data shards before communicating the updated model parameters back to the central server for averaging.
This iterative process is carried out for a total of 2000 and 200 communication rounds for CIFAR and Shakespeare respectively. The top-1 accuracies obtained are reported in Table 2.

Dataset	Top-1 accuracy
CIFAR-100	35.6
Shakespeare	32.7

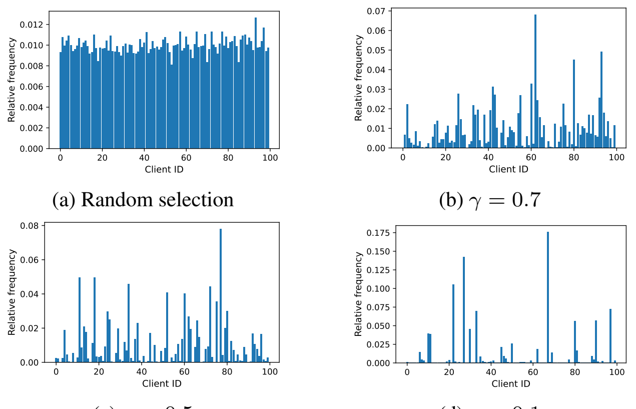
Table 2. Top-1 accuracies for CIFAR-100 and Shakespeare in the first FL baseline.

The federated learning results, as expected, are worse compared to the centralized baseline.
This discrepancy can be justified by several characteristics of federated learning. In particular, in the FL setting the training data are distributed across multiple clients, and each client only has access to a subset of data. Moreover, the averaging of the model parameters coming from different nodes can result in suboptimal updates [7].

4.4. The impact of client participation

To address another key aspect of the federated learning scenario, we aim to demonstrate how varying the selection probabilities of participants impacts overall performance. To simulate skewed sampling, we assign probabilities to the clients based on the Dirichlet distribution parameterized by the hyperparameter γ. The parameter γ controls the severity of the skewness: lower values of it result in higher heterogeneity in client sampling, as shown in figure 3. Notice that in the experiments within this subsection the clients present an IID distribution.
The results for CIFAR-100 are reported in figure 4 . In the plot, the *random selection* line represents the accuracy obtained with *uniform participation*, where all the nodes have the same probability to be selected. This scenario corresponds to the one in which $\gamma \to \infty$. The performance appears to scale linearly with the parameter γ: as γ increases, so does the test accuracy. Additionally, the curve representing random client participation reaches a plateau at a higher round compared to other settings. In the extreme case, when $\gamma = 0.1$, the accuracy ceases to improve after 500 rounds, whereas with random selection it continues to

improve up to 1300 rounds.
In contrast, for the Shakespeare dataset, the participation appears to have only a slight impact on the model's performance, as shown in Figure 5.



(a) Random selection (b) $\gamma = 0.7$

(c) $\gamma = 0.5$ (d) $\gamma = 0.1$

Figure 3. Clients' probabilities of being selected according to values drawn from a Dirichlet distribution with parameter γ

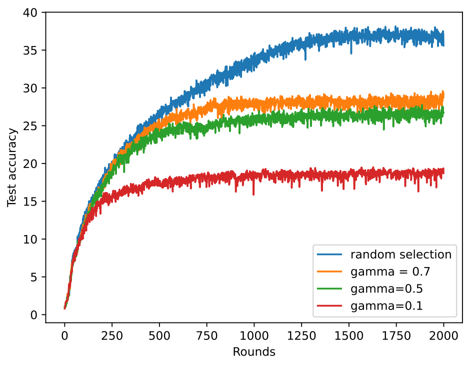


Figure 4. Comparison of the performance on CIFAR-100 with respect to different values of the parameter γ for the Dirichlet distribution.

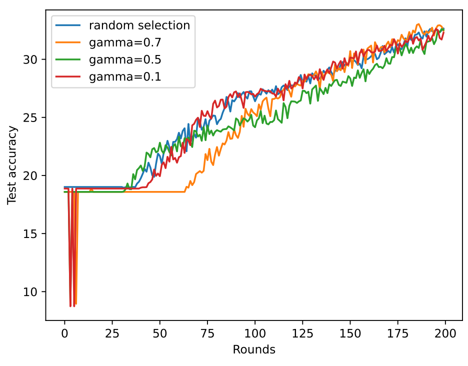


Figure 5. Comparison of the performance on Shakespeare with respect to different values of the parameter γ for the Dirichlet distribution.

4.5. Simulation of heterogenous scenarios

For the purpose of analysing the behaviour of FedAvg in a non-IID setting, we fixed $K = 100$ (number of clients) and $C = 0.1$ (fraction of clients participating in each round).
We reproduce several non-IID shardings of the CIFAR-100 training set by varying the number of different labels each local dataset has (N_C).
Each participant is configured to contain approximately the same number of records, and within each client, there is approximately the same number of records for each class. For Shakespeare dataset, a non-IID splitting of the data is already provided by Leaf, and we do not apply any sharding function for selecting the number of labels. The algorithm is tested varying the number of local steps J among $\{4, 8, 16\}$. To ensure a fair comparison, for each setting of J the number of training rounds is scaled accordingly and set to 2000, 1000 and 500 respectively for CIFAR-100 and to 200, 100 and 50 for Shakespeare.
The results of the simulations are reported in Table 3. For CIFAR-100, the performance in general does not improve with a larger value of J. In fact, increasing the number of local steps worsens performance for small N_C values, likely due to the gradient computation being heavily dependent on the local dataset, which only contains N_C labels. The only scenario where increasing local steps (J) appears beneficial is when N_C is set to 50. This suggests that with a small number of samples per class but a larger number of classes, more local steps can be advantageous. However, surprisingly, for $N_C = 50$ we obtain poorer results compared to $N_C = 10$. For $N_C = 1$, no improvement is observed.
Regarding Shakespeare, as illustrated in Figure 6, clients exhibit a similar distribution of labels due to their varying roles in different Shakespearean works. This consistency is a probable reason why the results, shown in Table 3, are

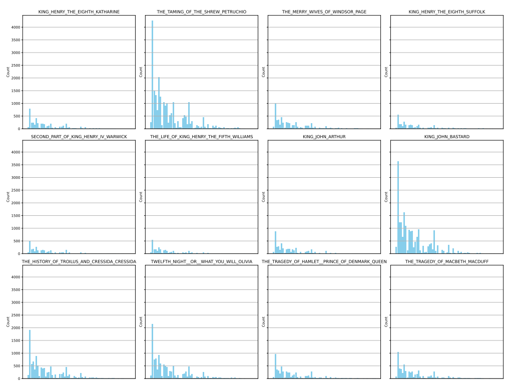


Figure 6. Non-IID label distribution of 12 random clients in Shakespeare dataset provided by Leaf.

significantly better than those for CIFAR-100.
We use the preprocessing function used in [1], which consists of selecting a sample of 2000 elements for each participant, and partially reduces the original difference among the clients' distributions.
We try different values for the number of local steps performed by each node, similarly to the image-classification task. Also in this case, the value of J does not help improving the accuracy.

Number of labels	**Number of local steps**		
	$J = 4$	$J = 8$	$J = 16$
$N_C = 50$	11.1	10.9	11.3
$N_C = 10$	23.4	22.2	20.4
$N_C = 5$	24.8	20.4	14.3
$N_C = 1$	1.0	1.0	1.0
Shakespeare	31.8	25.1	22.9

Table 3. Top-1 accuracies for CIFAR-100 and Shakespeare, for different number of local steps J and varying, only for CIFAR-100, the number of distinct class labels in each client N_C

4.6. Hybrid approaches experiments

In order to address the non-IID distribution of the data among clients, we repeat the experiments described in 4.5 using standard FedRep and the variations of the method proposed by us. The results are collected in Table 4.
For this task, we focus exclusively on the CIFAR-100 dataset, as it is the only dataset where we can easily control the number of classes. Additionally, CIFAR-100 presents more challenges in the non-IID scenario.

4.6.1 FedRep results

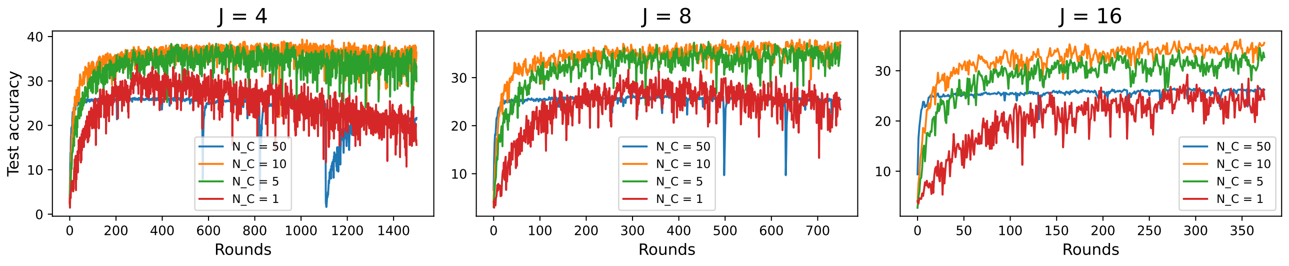
We firstly run basic FedRep method with the same hyperparameters as in the FedAvg approach. We apply our sharding function selecting $N_C = \{1, 5, 10, 50\}$ and we choose as client selection the random method, selecting 10% of clients at each round. Furthermore, we run the method scaling the number of global rounds R with respect to J as we do for FedAvg. Accuracy curve requires a certain number of rounds before starting increasing, which is inversely proportional to J. Nevertheless, at the end of the R communication rounds, for all the values of N_C the test accuracy is higher when $J = 4$, suggesting that increasing the number of local steps leads to worse solution in terms of accuracy, as already observed for FedAvg.
The poor quality of the result is consistent with what anticipated in section 2.3. Of course, FedRep is not able to reach reasonable results with $N_C = 1$ since it is performing worse than FedAvg in the specified number of rounds.

4.6.2 Pre-trained model and FedAvg2Rep results

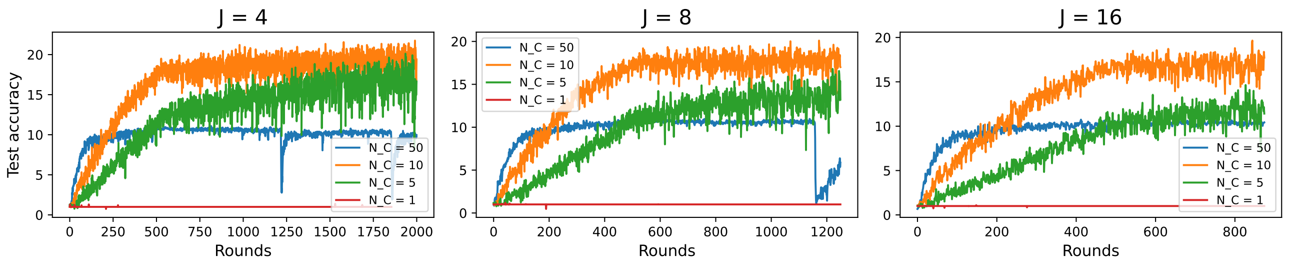
In our proposal, we train our global model using two specific methods before applying FedRep, as outlined in Section 2.3. To implement the first method aimed at improving FedRep's performance, we train FakeLeNet-5 on Tiny ImageNet for 200 epochs, maintaining the hyperparameters used in the centralized approach and reaching a test accuracy value of 34.7%. The resulting weights are then used as the starting global weights in the federated setting. We apply FedRep for various combinations of the number of classes per client N_C and local steps J, using a number of rounds given by $R = 1500(4/J)$. In this phase, we reduce the total number of rounds, considering the warm-up as beneficial for the training process.
For the FedAvg2Rep warm-up, we conduct $R_1 = 500$ rounds using standard FedAvg with $J = 4$ local steps, as this value yields the best accuracy across all heterogeneous scenarios discussed in Section 4.5.
Subsequently, we ran $R_2 = 1500(4/J)$ rounds with FedRep to achieve comparable results across methods. This approach ensures that the number of FedRep rounds matches those used for the training scheme leveraging the pre-trained model.
Figure 7 displays the test accuracy at each round for FedRep with a pre-trained model and for FedAvg2Rep, while Table 4 presents the final accuracies at the end of the training process.
Pretraining on the ImageNet dataset returns superior results compared to the other training schemes across all N_C values. It shows a good accuracy value even for $N_C = 1$, but from Figure 7a we can easily see how, for $J = 4$ and $J = 8$, the accuracy for single-class clients initially peaks but then tends to decline. Moreover, we observe that increasing the number of local steps J using this training scheme leads to improved performance, showing a corresponding increase, which is not true for FedAvg and FedRep alone.
FedAvg2Rep significantly improves the global model's accuracy in the initial phase, with the accuracy curve continuing to rise gradually with the FedRep method, particularly for $N_C = 5$ and $N_C = 10$, with all the values of J (Figure 7b).
The final results surpass the basic FedRep values and are slightly lower than the FedAvg accuracy (see Table 2 for FedAvg values). However, FedRep2Avg does not appear to address the issues encountered with $N_C = 1$.
Consistent with the findings in Table 2, also in this case the optimal value of J seems to be 4, and a higher number of local steps does not positively affect the accuracy values.

	$N_C = 1$			$N_C = 5$			$N_C = 10$			$N_C = 50$		
	$J = 4$	$J = 8$	$J = 16$	$J = 4$	$J = 8$	$J = 16$	$J = 4$	$J = 8$	$J = 16$	$J = 4$	$J = 8$	$J = 16$
FedRep	1.0	1.0	1.1	6.3	5.2	4.1	14.1	9.7	5.7	8.6	7.4	7.0
Pre-trained	15.6	23.4	24.4	29.9	36.7	32.7	36.4	37.4	35.6	21.7	24.8	26.4
FedAvg2Rep	1.0	1.0	1.0	16.7	13.2	11.4	17.9	17.1	17.8	10.0	5.9	10.4

Table 4. Accuracies on the test set achieved at the end of training using various training schemes and hyperparameter configurations.



(a) Starting from a pre-trained model



(b) FedAvg2Rep

Figure 7. Test accuracies using FedRep starting from a pre-trained model and FedAvg2Rep, for varying local steps in the FedRep phase and different numbers of classes per client. The number of rounds is adjusted accordingly.

It is important to notice that, although FedAvg outperforms FedAvg2Rep, the new method enables clients to create personalized models, performing well on both global and local data.

5. Conclusion

In this work, we analyzed the main challenges in federated learning and proposed solutions to address them. We began by implementing a centralized model and then transitioned to a federated learning architecture, simulating the key scenarios for this setup. We proposed a solution to the key problems, particularly those related to the non-IID distribution of data among clients, by introducing a warm-up phase for the global model before applying the existing FedRep algorithm, which focuses on the personalization of the local models. We improved the results obtained by FedRep by applying FedAvg2Rep, which involves using FedAvg for the warm-up phase. Furthermore, the use of a pre-trained model for the same purpose proved to be very effective in highly heterogeneous environments, compared to standard FedRep. Generally, our proposal proved to be an effective trade-off between global accuracy and personalized models, enabling participants to learn customized models while still achieving reasonable improvements on a global scale.

References

[1] Acar et al. Federated learning based on dynamic regularization, 2021. 4, 7

[2] Caldas et al. Leaf: A benchmark for federated settings, 2019. 4, 5

[3] Collins et al. Exploiting shared representations for personalized federated learning. 2023. 2

[4] Chen et al. Privacy and fairness in federated learning: on the perspective of trade-off. 2023. 1

[5] Hsu et al. Federated visual classification with real-world data distribution, 2020. 4

[6] Le et al. Tiny imagenet visual recognition challenge. 2015. 4

[7] McMahan et al. Communication-efficient learning of deep networks from decentralized data. 2023. 1, 5

[8] Reddi et al. Adaptive federated optimization, 2021. 4

[9] Krizhevsky. Learning multiple layers of features from tiny images. 2009. 4