Federated Learning: where machine learning and data privacy can coexist

INTRODUCTION

Federated learning is a distributed form of machine learning that involves multiple devices. Each client is trained using different data and develops its own model to perform a specific task. The clients then send their templates to a monitored server. The centralized server draws on each of these models to create a hybrid model, which offers performance over another model alone. The central server then sends this hybrid model to each of the clients. The whole process is then assigned, with each iteration leading to model updates that ultimately improve system performance.

This approach allows us to build machine learning systems without direct access to training data, which remain in their original location, ensuring privacy and reducing communication costs. Federated learning is ideal for smartphones and edge hardware, healthcare services, privacy-sensitive use cases, and industrial applications such as predictive maintenance.

In this work, we will train an image classifier trained on the CIFAR10 dataset with federated learning, we will initially try to understand what are the right parameters to initialize the system and we will test various aggregation algorithms of the client models by simulating some data distributions that can be similar to those in the real world.

RELATED WORK

One of the most important and most addressed problems in federated learning is that of the statistical heterogeneity of the data distributions. Many authors focus on ideal datasets, with independent and identically distributed datasets (iid) in order to provide an upper bound for more realistic dataset splits that take into account the non-identicalness of the clients [12]. When dealing with realistic data, many factors influence the likeliness of the data among the users, like geographical location, habits, health conditions or personal preferences. Hsu et al. [14] propose a parametric CIFAR10 split that generalize the iid scenario. In [4], another approach is considered: the kind of data which can be found on each client is highly imbalanced and between "majority" and "minority" groups.

Local models can vary significantly from each other. The first approach to the aggregation of the local models has been provided by McMahan et al. [13] with the FederatedAveraging algorithm (FedAvg), later expanded with the introduction of a momentum to the server (FedAvgM) [14]. FedProx [7] adds a proximal term to account for statistical heterogeneity by limiting the impact of local updates. Scaffold [9] corrects the gradient of each local update to reduce the variance..

METHODS

The purpose of the tests was to compare the behaviour of the various networks and algorithms in different distributions of the dataset. For our experiments, Client and Server classes have been implemented that encapsulate all the local data necessary to be able to train and test the model. In particular, each Client has its own network, its own training set and its own test set.

3.1 Distributions

Several distributions of training sets have been generated starting from the complete dataset CIFAR10 [1].

The first is a simple distribution that we will call "non_iid" [2], where each client has a random range of classes with a minimum of 1 and a maximum of 7 classes. Samples per class for each client are fully balanced, and extracted without re-entry.

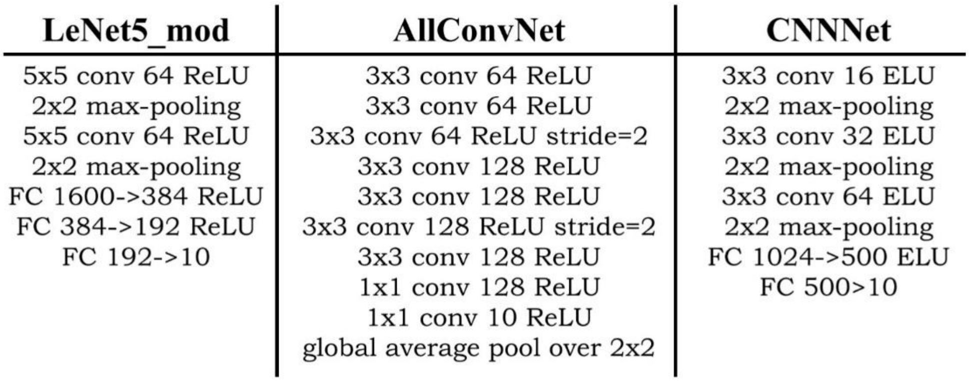
We then have two other distributions that can better represent a real-world scenario: Dirichlet distribution proposed in [3] and multimodal distribution [4]. The second distribution divides the classes into two groups called "modes" (in our case they are animals and vehicles) to which each client belongs and creates partitions based on two parameters: the number of classes per client (Z) and the percentage of presence of the first group within the population (RATIO).

The test sets of each client are generated through an extraction with reintegration from the original test set to obtain a balanced dataset with the same classes.

Appendix A.1 provides a graphical representation of the various distributions.

3.2 Models

For the choice of networks, we looked for architectures suitable for CIFAR10 and at the same time light and fast. A modified LeNet5 described in [3], the All-Convolutional-Network described in [5] modified by reducing the number of output channels for each convolution layer by a factor of 2/3 and finally a simple CNN network that uses Elu as an activation function [6].



LeNet5_mod	AllConvNet	CNNNet
5x5 conv 64 ReLU	3x3 conv 64 ReLU	3x3 conv 16 ELU
2x2 max-pooling	3x3 conv 64 ReLU	2x2 max-pooling
5x5 conv 64 ReLU	3x3 conv 64 ReLU stride=2	3x3 conv 32 ELU
2x2 max-pooling	3x3 conv 128 ReLU	2x2 max-pooling
FC 1600->384 ReLU	3x3 conv 128 ReLU	3x3 conv 64 ELU
FC 384->192 ReLU	3x3 conv 128 ReLU stride=2	2x2 max-pooling
FC 192->10	3x3 conv 128 ReLU	FC 1024->500 ELU
	1x1 conv 128 ReLU	FC 500>10
	1x1 conv 10 ReLU	
	global average pool over 2x2	

Figure 1: Architectures of the chosen networks

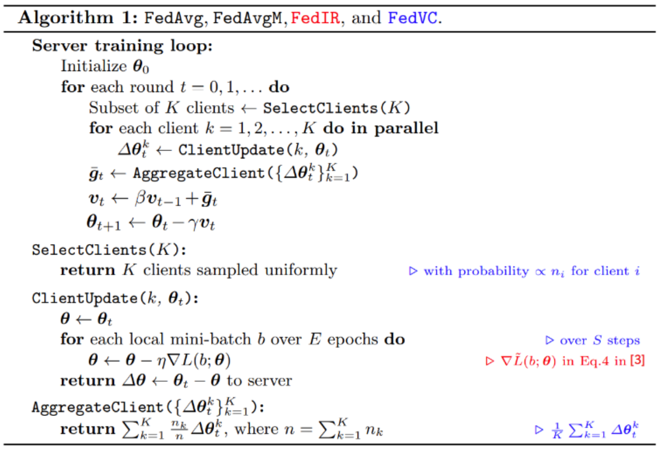
3.3 Algorithms

The main algorithms we have tested and compared are FedAvg and FedAvgM each with the FedIR and FedVC variants described in [3].

FedAvg is the best known and simplest federated learning algorithm and consists in applying a weighted average to the updates of the selected client models.

FedAvgM in addition makes use of a momentum when optimizing the global model. The β factor is called server momentum and is zero on FedAvg.

FedIR and FedVC are two variants of the algorithms just described. The first applies a weighting of the objective function of each client based on the relationship between an ideal distribution of the data and the current one. FedVC instead keeps each client's datasets balanced by extracting a predefined number NVC of images before training.



```
Algorithm 1: FedAvg, FedAvgM, FedIR, and FedVC.
Server training loop:
    Initialize $\boldsymbol{\theta}_0$
    for each round $t = 0, 1, \ldots$ do
        Subset of $K$ clients ← SelectClients($K$)
        for each client $k = 1, 2, \ldots, K$ do in parallel
            $\Delta\boldsymbol{\theta}_t^k$ ← ClientUpdate($k$, $\boldsymbol{\theta}_t$)
        $\bar{\boldsymbol{g}}_t$ ← AggregateClient($\{\Delta\boldsymbol{\theta}_t^k\}_{k=1}^K$)
        $\boldsymbol{v}_t \leftarrow \beta\boldsymbol{v}_{t-1} + \bar{\boldsymbol{g}}_t$
        $\boldsymbol{\theta}_{t+1} \leftarrow \boldsymbol{\theta}_t - \gamma\boldsymbol{v}_t$
SelectClients($K$):
    return $K$ clients sampled uniformly        ▷ with probability $\propto n_i$ for client $i$
ClientUpdate($k$, $\boldsymbol{\theta}_t$):
    $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_t$
    for each local mini-batch $b$ over $E$ epochs do        ▷ over $S$ steps
        $\boldsymbol{\theta} \leftarrow \boldsymbol{\theta} - \eta\nabla L(b; \boldsymbol{\theta})$        ▷ $\nabla\tilde{L}(b; \boldsymbol{\theta})$ in Eq.4 in [3]
    return $\Delta\boldsymbol{\theta} \leftarrow \boldsymbol{\theta}_t - \boldsymbol{\theta}$ to server
AggregateClient($\{\Delta\boldsymbol{\theta}_t^k\}_{k=1}^K$):
    return $\sum_{k=1}^K \frac{n_k}{n}\Delta\boldsymbol{\theta}_t^k$, where $n = \sum_{k=1}^K n_k$        ▷ $\frac{1}{K}\sum_{k=1}^K \Delta\boldsymbol{\theta}_t^k$
```

Other aggregation algorithms developed for non-iid distributions such as FedProx [7], FedNova [8] and SCAFFOLD [9] have also been implemented and tested.

FedProx introduces an additional regularization term, weighted by an hyperparameter μ, in the local objective function to limit the distance between the local model and the global model. FedNova instead, during the aggregation phase, weighs the local updates of each client according to their number of optimization steps. Finally, SCAFFOLD introduces the variance between the parts and applies the variance reduction technique. The pseudocode of each of these algorithms is detailed in Appendix A.2.

3.4 Metrics

The metric chosen to evaluate the model is relative accuracy, defined as the relationship between two accuracies:

- Weighted accuracy: a weighted average of the accuracies calculated by each client on its train_set, where the weights correspond to the number of samples observed by each client.
- Centralized accuracy: the performance of the network in a classic centralized system.

EXPERIMENTS

In the preliminary phase, some parameters were chosen to be fixed.

The total number of clients representing the population is fixed at 100 for any test and standard values of learning rate, momentum, weight decay and batch size have been chosen for the local training of the clients 0.1, 0.9, 4e-4 and 64 respectively.

Other parameters that have been set are NVC = 256 for FedVC and μ = 0.3 for FedProx.

4.1 Ablation studies

A FedAvg ablation study on non_iid distribution with CNNNet was conducted to observe performance behavior, varying the number of rounds, number of clients selected for rounds and local epochs.

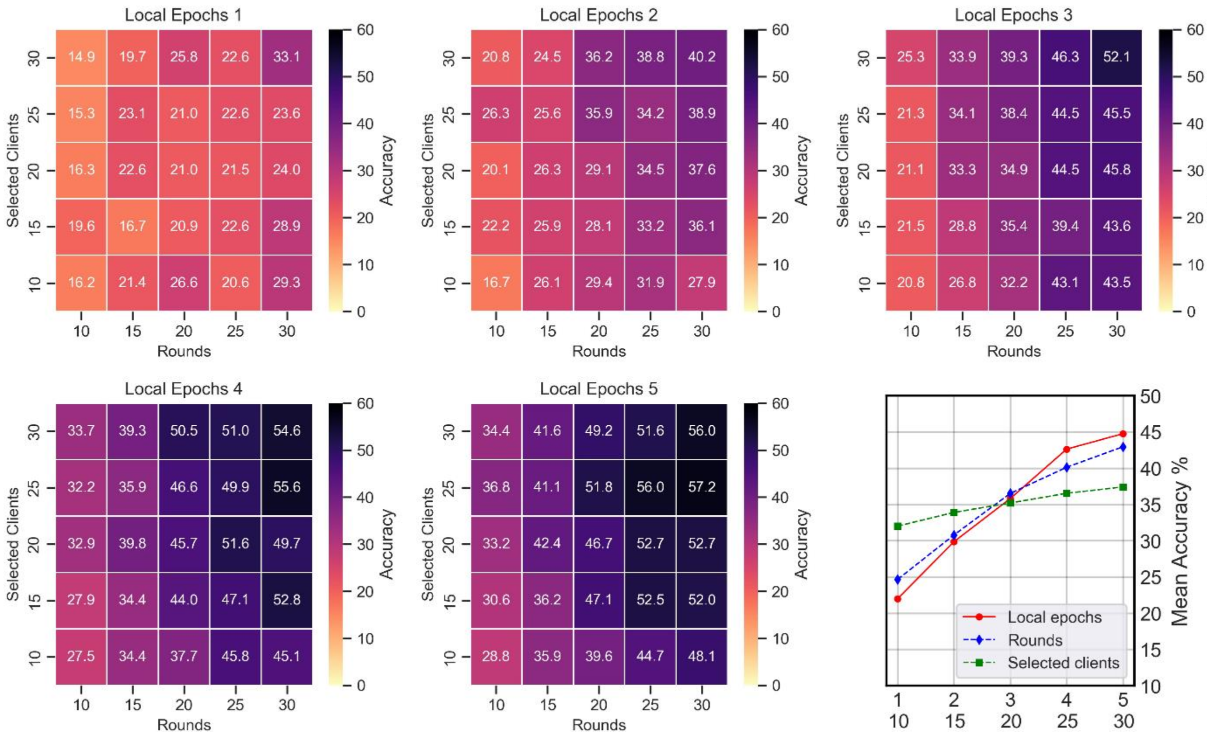
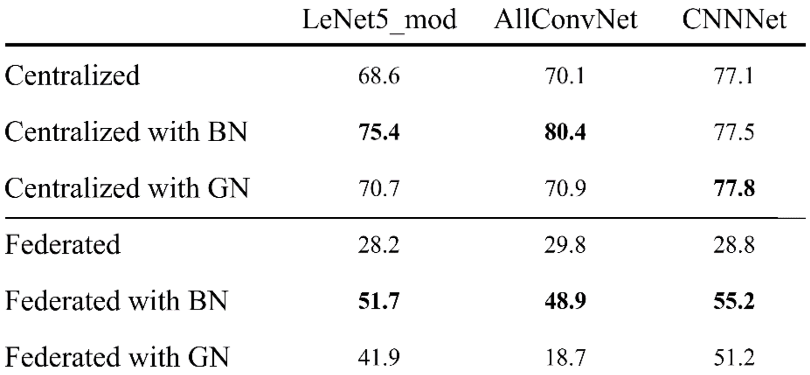


Figure 2: The number of local epochs, the number of selected clients and the rounds vary in a range of 5 elements each. The graph at the bottom right shows the average of all the accuracies we get if we select the corresponding value for each parameter.

As it was logical to expect, these are all parameters directly proportional to the accuracy of the model. In particular, from the graph of the average accuracies we note that in this case the local epochs and the number of rounds have a higher variance than the number of clients selected for rounds, so they could be more influential on the performance of the tests. We wanted to keep both the number of selected clients and the number of rounds for each test high by setting their value at 25, while we decided to keep the number of local epochs low at 3 since this is a parameter that could overfit some clients and create instability in the global model.

Another small preliminary study was conducted to see how performance can change using batch normalization and group normalization. The following table shows the absolute accuracies of the various networks used both in the centralized standard system and in a federated system using the same parameters chosen previously.



	LeNet5_mod	AllConvNet	CNNNet
Centralized	68.6	70.1	77.1
Centralized with BN	**75.4**	**80.4**	77.5
Centralized with GN	70.7	70.9	**77.8**
Federated	28.2	29.8	28.8
Federated with BN	**51.7**	**48.9**	**55.2**
Federated with GN	41.9	18.7	51.2

Figure 3: Centralized tests were performed on the entire complete dataset by setting the number of epochs to 30. Federated tests were performed with FedAvg and a non_iid distribution, local epochs 3, selected clients and rounds 25. BN and GN are the tests with the networks to which a normalization layer (batch and group respectively) has been added between each convolutional layer and the respective activation function.

In our tests we have observed how in general applying a batch normalization greatly increases both the accuracy of the model and the speed of convergence. For this reason, we decided to run the tests using batch normalization on each network.

Now we are going to summarize the results obtained on the Dirichlet distribution and on the Multimodal distribution with all the parameters discussed above and all the aggregation algorithms implemented: FedAvg, FedAvgM, FedProx, FedNova and SCAFFOLD.

4.2 Dirichlet distribution

The split to generate the distribution was taken from [10] which contains the indexes to be associated with 100 clients for each alpha value. The performances were measured with the Relative Accuracy described in paragraph 3.2, i.e., the Weighted Accuracy was calculated and related to the centralized accuracy of the corresponding model. The higher this measure is, the more the distribution/algorithm pair approaches the performance of a classic centralized system.

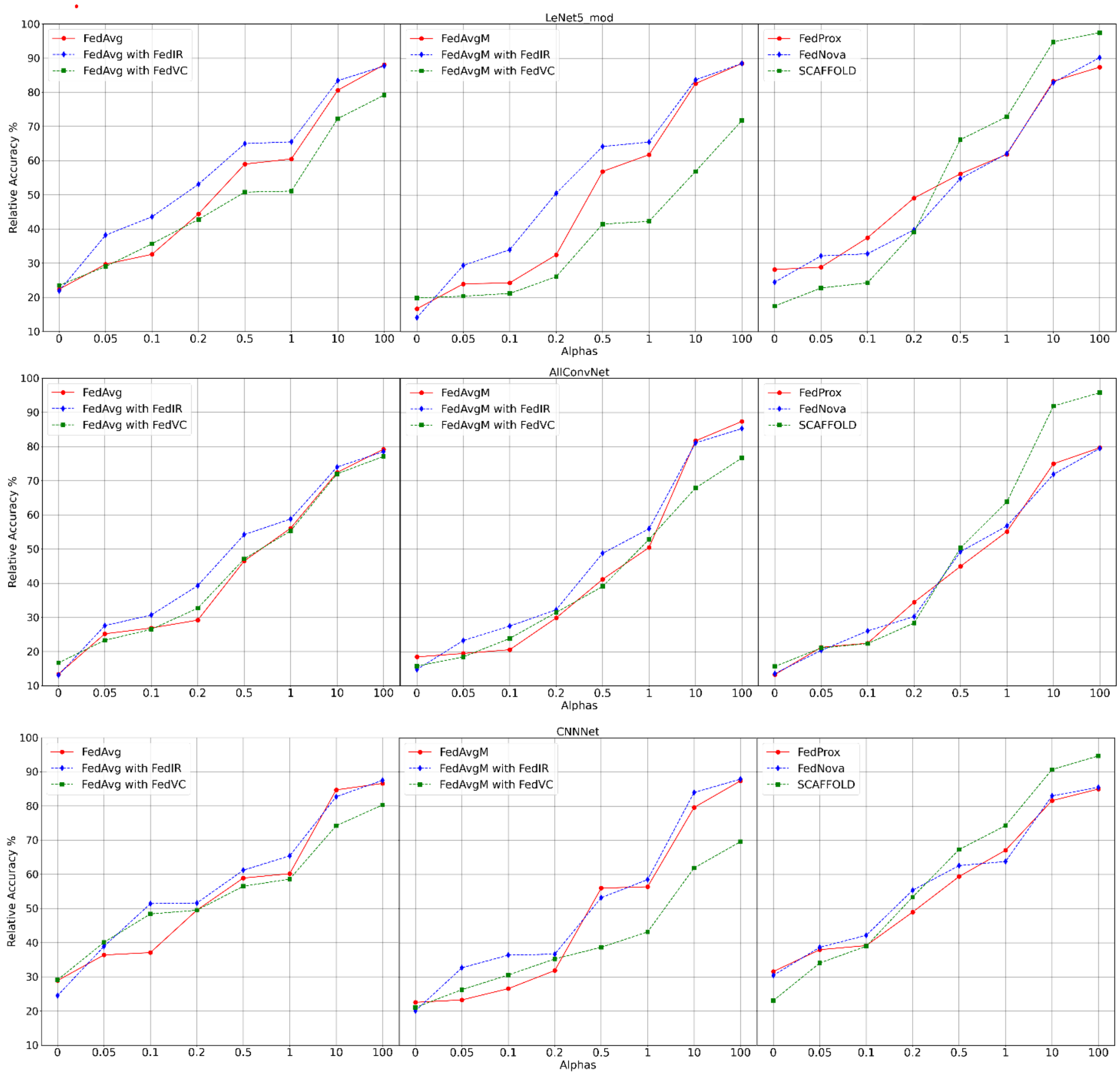


Figure 4: Relative accuracy as a function of alpha for each model and algorithm.

We see from the first two columns, how FedIR's reweighting technique provides better overall performance at small alpha values. Conversely SCAFFOLD seems to suffer a little with small alpha values, but when the datasets start to become more homogeneous it returns higher accuracy values. With alpha=100 SCAFFOLD behaved practically like the centralized system on all three networks.

4.3 Multimodal distribution

The cifar_multimodal_noniid() function divides the client population based on the ratio into two subpopulations and randomly assigns Z labels to each client. After that, an image for each client is fished without re-insertion until the dataset is exhausted. This split does not take into account the imbalance in the number of data in the training sets of the clients. For example, with a very low ratio, a single client could receive all the images of animals.

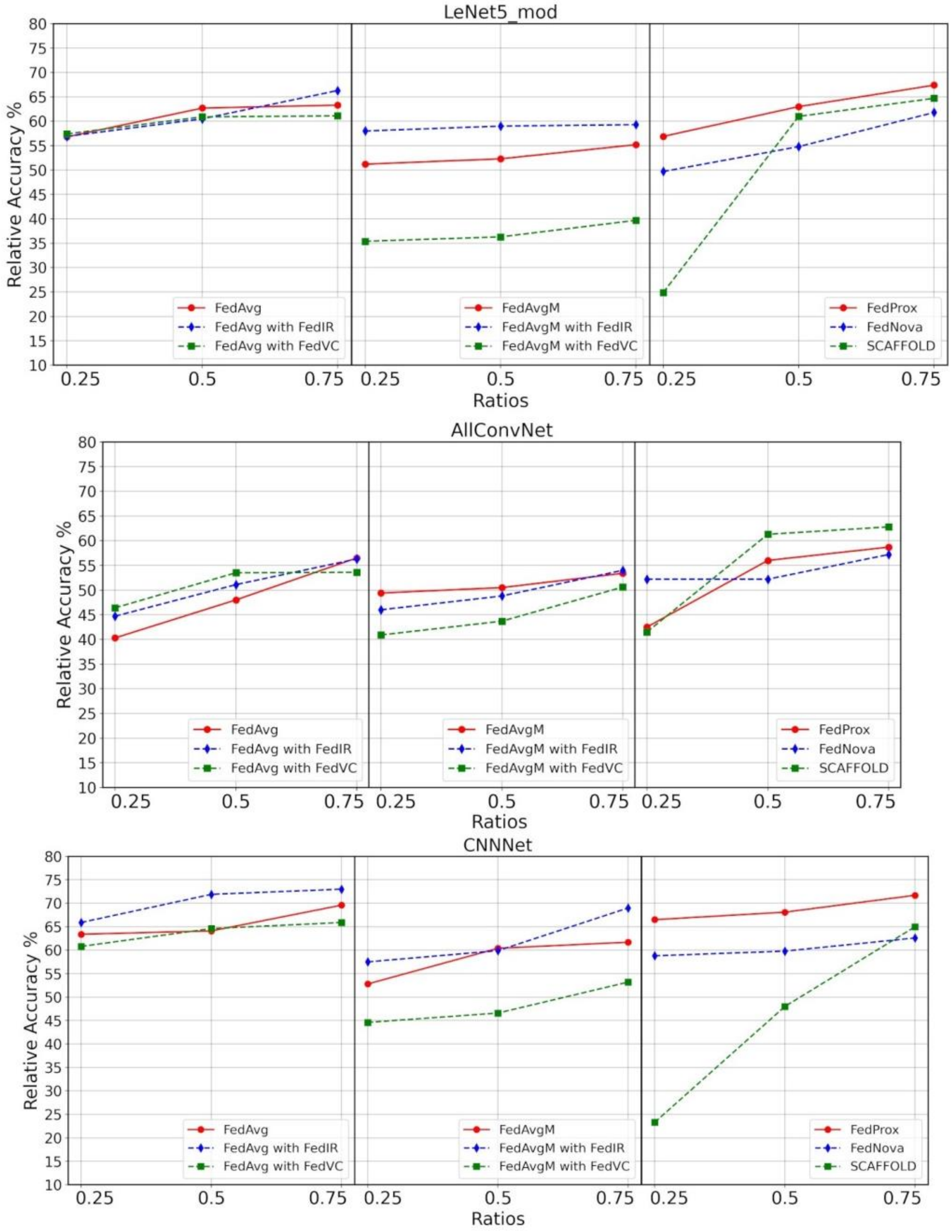


Figure 5: Relative accuracy as a function of the Ratio variable with Z = 3.

The performance in general increases slightly with a higher ratio probably because the number of classes of animals is greater than those of vehicles for which a greater balance is created in terms of number of images per client. CNNNet in this case performed better than the other networks especially with FedAvg with FedIR and FedProx.

CONCLUSIONS

In this work we compared different types of aggregation algorithms and different distributions on federated learning showing how crucial it is for this type of learning how data is distributed among the various nodes of the network. Of course, the combinations of parameters used are partial. Further research could consist in using and testing deeper pre-trained networks such as GoogleNet or ResNet18, or carrying out a preliminary Grid-Search tuning phase on other ranges of hyperparameters such as local epochs, learning rate (both of the client and of the server), batch size.

References

1. CIFAR-10 and CIFAR-100 datasets (toronto.edu)
2. **https://towardsdatascience.com/preserving-data-privacy-in-deep-learning-part-2-6c2e9494398b**
3. [2003.08082] Federated Visual Classification with Real-World Data Distribution (arxiv.org)
4. [2008.05687] WAFFLe: Weight Anonymized Factorization for Federated Learning (arxiv.org)
5. [1412.6806] Striving for Simplicity: The All Convolutional Net (arxiv.org), https://github.com/StefOe/all-conv-pytorch/blob/master/allconv.py
6. **https://github.com/simoninithomas/cifar-10-classifier-pytorch/blob/master/PyTorch%20Cifar-10%20Classifier.ipynb**
7. [1812.06127] Federated Optimization in Heterogeneous Networks (arxiv.org)
8. [2007.07481] Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization (arxiv.org)
9. [1910.06378] SCAFFOLD: Stochastic Controlled Averaging for Federated Learning (arxiv.org)
10. **https://arxiv.org/abs/2102.02079**
11. Xtra-Computing/NIID-Bench: Federated Learning on Non-IID Data Silos: An Experimental Study (ICDE 2022) (github.com)
12. [1806.00582] Federated Learning with Non-IID Data (arxiv.org)
13. [1602.05629] Communication-Efficient Learning of Deep Networks from Decentralized Data (arxiv.org)
14. [1909.06335] Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification (arxiv.org)

APPENDIX

A.1 Distributions

The following figure shows the 10 client training sets in a population of 100 for each distribution.

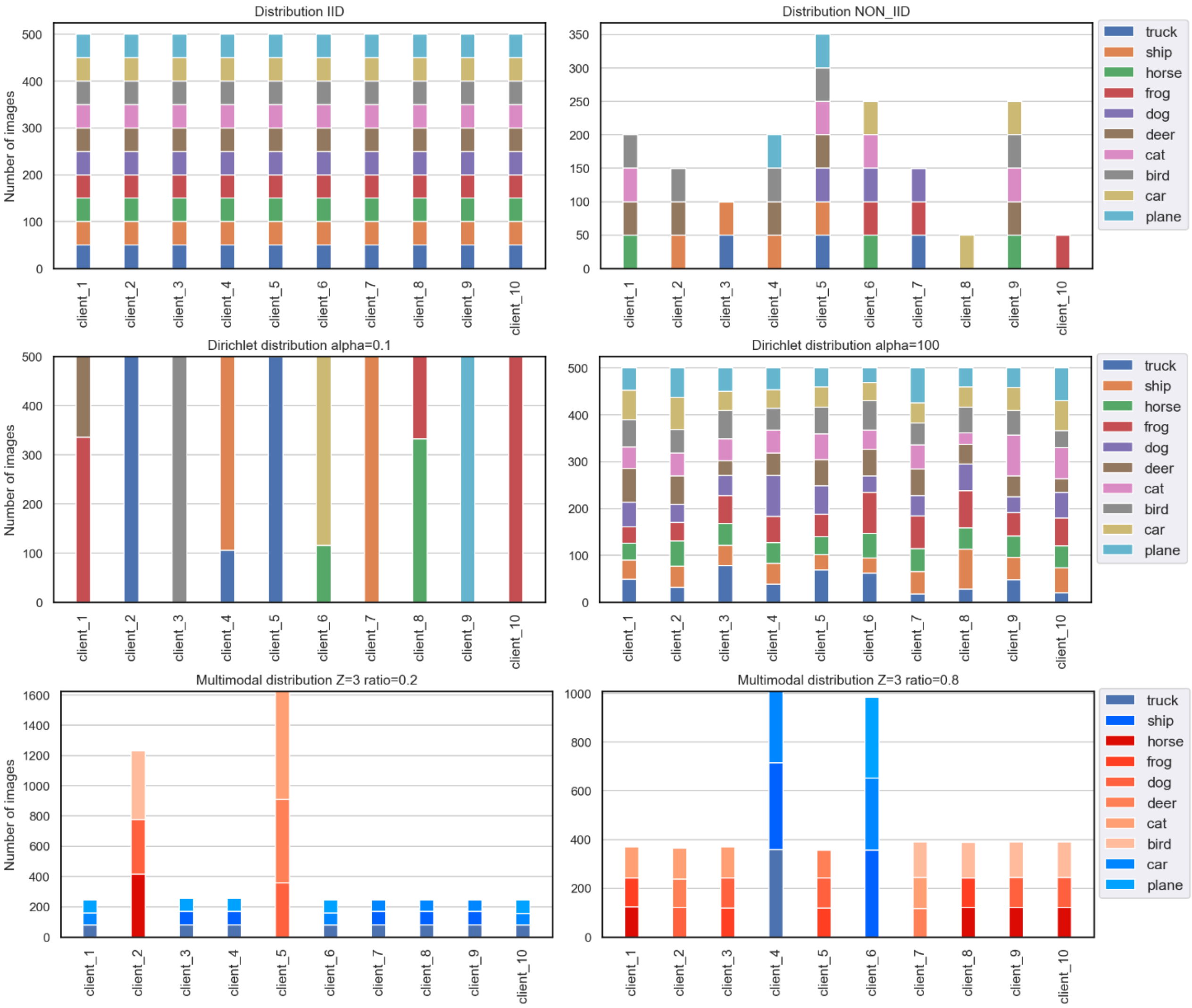


Figure 6: Training set of clients for each type of distributions.

The non_iid distribution [2] and Dirichlet distributions [3] create balanced datasets in the number of images per client but the classes are distributed unevenly. The first assigns a random number of classes per client up to a maximum of 7, the second instead controls the distribution of the classes based on an alpha parameter.

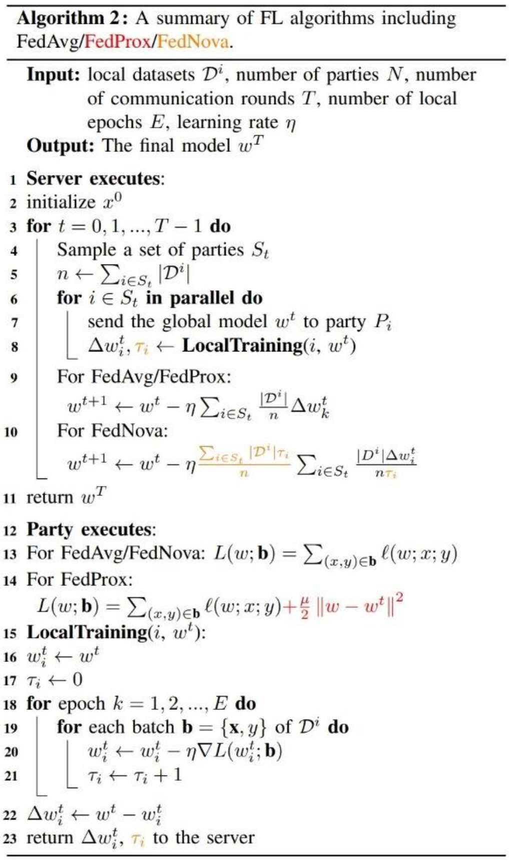
Multimodal distribution [4] requires the population to be divided into two groups: animals and vehicles. The variable Z constitutes the number of labels for each client while the "ratio" controls the percentage of presence of a class within the population. Other variants of this distribution involve keeping the number of images for each client balanced, or inserting noise within the training set of clients represented by elements belonging to the opposite group.

Each client also encapsulates its own test set generated from the original CIFAR10 test set which maintains the exact same distribution of the corresponding test set but with 80% fewer images.

A.2 FedProx, FedNova & SCAFFOLD

FedProx [7] improves the local objective based on FedAvg.
It directly limits the size of local updates. Specifically, as shown in Line 14 of Algorithm 2, it introduces an additional regularization term in the local objective function to limit the distance between the local model and the global model. This is a straightforward way to limit the local updates so that the averaged model is not so far from the global optima. If µ is too small, then the regularization term has almost no effect. If µ is too big, then the local updates are very small and the convergence speed is slow.

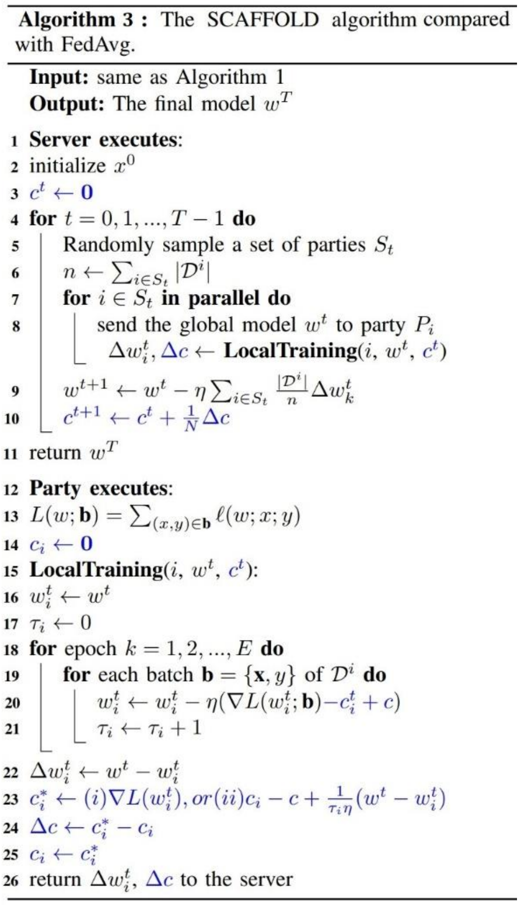
FedNova [8] improves FedAvg in the aggregation stage. It considers that different parties may conduct different numbers of local steps (i.e., the number of mini-batches in the local training) each round. This can happen when parties have different computation power given the same time constraint or parties have different local dataset size given the same number of local epochs and batch size. Intuitively, the parties with a larger number of local steps will have a larger local update, which will have a more significant influence on the global updates if simply averaged. Thus, to ensure that the global updates are not biased, FedNova normalizes and scales the local updates of each party according to their number of local steps before updating the global model (Line 10 of Algorithm 2). FedNova also only introduces lightweight modifications to FedAvg, and negligible computation overhead when updating the global model.



Algorithm 2: A summary of FL algorithms including FedAvg/FedProx/FedNova.

```
Input: local datasets $\mathcal{D}^i$, number of parties $N$, number
       of communication rounds $T$, number of local
       epochs $E$, learning rate $\eta$
Output: The final model $w^T$
1  Server executes:
2  initialize $x^0$
3  for $t = 0, 1, ..., T-1$ do
4      Sample a set of parties $S_t$
5      $n \leftarrow \sum_{i \in S_t} |\mathcal{D}^i|$
6      for $i \in S_t$ in parallel do
7          send the global model $w^t$ to party $P_i$
8          $\Delta w_i^t, \tau_i \leftarrow$ LocalTraining($i$, $w^t$)
9      For FedAvg/FedProx:
        $w^{t+1} \leftarrow w^t - \eta \sum_{i \in S_t} \frac{|\mathcal{D}^i|}{n} \Delta w_k^t$
10     For FedNova:
        $w^{t+1} \leftarrow w^t - \eta \frac{\sum_{i \in S_t} |\mathcal{D}^i| \tau_i}{n} \sum_{i \in S_t} \frac{|D^i| \Delta w_i^t}{n \tau_i}$
11 return $w^T$
12 Party executes:
13 For FedAvg/FedNova: $L(w; \mathbf{b}) = \sum_{(x,y) \in \mathbf{b}} \ell(w; x; y)$
14 For FedProx:
    $L(w; \mathbf{b}) = \sum_{(x,y) \in \mathbf{b}} \ell(w; x; y) + \frac{\mu}{2} \|w - w^t\|^2$
15 LocalTraining($i$, $w^t$):
16 $w_i^t \leftarrow w^t$
17 $\tau_i \leftarrow 0$
18 for epoch $k = 1, 2, ..., E$ do
19     for each batch $\mathbf{b} = \{\mathbf{x}, y\}$ of $\mathcal{D}^i$ do
20         $w_i^t \leftarrow w_i^t - \eta \nabla L(w_i^t; \mathbf{b})$
21         $\tau_i \leftarrow \tau_i + 1$
22 $\Delta w_i^t \leftarrow w^t - w_i^t$
23 return $\Delta w_i^t$, $\tau_i$ to the server
```

SCAFFOLD [9] models non-IID as introducing variance among the parties and applies the variance reduction technique. It introduces control variates for the server (i.e., c) and parties (i.e., ci), which are used to estimate the update direction of the server model and the update direction of each client. Then, the drift of local training is approximated by the difference between these two update directions. Thus, SCAFFOLD corrects the local updates by adding the drift in the local training (Line 20 of Algorithm 3). SCAFFOLD proposes two approaches to update the local control variates (Line 23 of Algorithm 3), by computing the gradient of the local data at the global model or by reusing the previously computed gradients. The second approach has a lower computation cost while the first one may be more stable. In our tests we used the second approach.



Algorithm 3 : The SCAFFOLD algorithm compared with FedAvg.

```
Input: same as Algorithm 1
Output: The final model $w^T$
1  Server executes:
2  initialize $x^0$
3  $c^t \leftarrow \mathbf{0}$
4  for $t = 0, 1, ..., T-1$ do
5      Randomly sample a set of parties $S_t$
6      $n \leftarrow \sum_{i \in S_t} |\mathcal{D}^i|$
7      for $i \in S_t$ in parallel do
8          send the global model $w^t$ to party $P_i$
            $\Delta w_i^t, \Delta c \leftarrow$ LocalTraining($i$, $w^t$, $c^t$)
9      $w^{t+1} \leftarrow w^t - \eta \sum_{i \in S_t} \frac{|\mathcal{D}^i|}{n} \Delta w_k^t$
10     $c^{t+1} \leftarrow c^t + \frac{1}{N} \Delta c$
11 return $w^T$
12 Party executes:
13 $L(w; \mathbf{b}) = \sum_{(x,y) \in \mathbf{b}} \ell(w; x; y)$
14 $c_i \leftarrow \mathbf{0}$
15 LocalTraining($i$, $w^t$, $c^t$):
16 $w_i^t \leftarrow w^t$
17 $\tau_i \leftarrow 0$
18 for epoch $k = 1, 2, ..., E$ do
19     for each batch $\mathbf{b} = \{\mathbf{x}, y\}$ of $\mathcal{D}^i$ do
20         $w_i^t \leftarrow w_i^t - \eta(\nabla L(w_i^t; \mathbf{b}) - c_i^t + c)$
21         $\tau_i \leftarrow \tau_i + 1$
22 $\Delta w_i^t \leftarrow w^t - w_i^t$
23 $c_i^* \leftarrow (i) \nabla L(w_i^t), or (ii) c_i - c + \frac{1}{\tau_i \eta}(w^t - w_i^t)$
24 $\Delta c \leftarrow c_i^* - c_i$
25 $c_i \leftarrow c_i^*$
26 return $\Delta w_i^t$, $\Delta c$ to the server
```

The FedNova and SCAFFOLD implementation was taken from [11] and adapted to our code.