AML Project

Federated Learning: where machine learning and data privacy can coexist

Alfredo Baldó Chamorro — Bastien Bertholom — Giuseppe Salvi

13 February 2022

1 Abstract

With the mass of data generated each day by multiple sources (i.e. IoT), machine learning has a bright future. Unfortunately, a huge part of this data is not usable due to their private nature regarding users. Federated learning is a machine learning framework aiming to guarantee this privacy by keeping the data locally on the device (i.e. mobile phones).

2 Introduction

Introduced by Google in 2016 [Kon+16], Federated Learning enables models to be iteratively trained in a decentralized way, on local devices. At each round, a subset of sources (also called *clients*) are selected to perform local training on their own data. Then, the centralized model aggregates the ensemble of updates to make the new global model and send back it to another subset of clients.

As a part of the Advanced Machine Learning course at Politecnico di Torino, this paper will cover the work on the replication of the experiment proposed by [HQB20] on the CIFAR10 dataset which consists of analyzing the effects that client's data distributions have on federated models. Especially distributions such as Non-Identical Class Distribution (each client has a different class distribution) and Imbalanced Client Sizes (each client has a different amount of data).

The work will cover the federated scenario using the standard FedAvg algorithm with multiple sets of parameters and evaluate the one with the greatest performance. Experiments with different settings such as data distribution, normalization layers and group normalization will be implemented. Finally, there will be a solution proposal (which will be a test) in order to solve one of the identified problems.

3 Related Work

This experiment has already been performed by [HQB20] on the iNaturalist-2017 dataset. They provided an analysis of the effect of learning with per-user data and proposed two new algorithms (FedVC, FedIR) and provided new large-scale datasets with per-user data.

A different algorithm (FedAvgM) has been proposed by [HQB19] using momentum on top of SGD to accelerate the network training. This approach changes the way the weights are updated to add momentum at the server side.

4 Methods

There are various steps in the building of the final algorithm. In this section the decisions that led to choose the values for the parameters and the choice for different algorithms implemented will be explained.

Federated learning applied in the real world, would take into account multiple devices with different characteristics (CPU capacity and speed...). The model and algorithm selection was chosen on the basis of a hypothetical implementation on different devices.

4.1 Problem setup

Federated Learning consists of a client and a server side. Considering the CIFAR10 dataset contains 50000 samples for training and 10000 samples for testing, choosing the number of clients K = 100 was thought as to be representing of a real-world scenario (averaging 500 images per client for training).

Moreover, in order to simulate real world data, a Delta variable was implemented into the code in order to add randomness regarding the distribution of the data among clients. The variable added some casualty to the quantity of data that each client would be seeing. For example, if Delta = 100, each of the 100 clients would have data between a minimum of 400 images and a maximum of 600. (50 000 / 100 = 500 +- 100).

To add up, the distribution of data was considered to be performed randomly. The variability Delta introduced was not substantial, and this was done intentionally. The problem of some client having way bigger amount of images than others (2-3 times) could have been solved with the creation of additional virtual clients and splitting the data among them. In terms of implementation this would be equal to increasing the number of clients from K to K + i, where i is the number of the virtual clients created.

Moreover in order to test the model, the accuracy on the training was computed as an average of the accuracy on the client's data. On the validation part, the accuracy was tested with the whole validation dataset on the server side.

Finally the number of clients C, was chosen based on performance. As the table 1, relates, the best performance is for C=0.4, however the number of clients increases significantly and hence computational time. For this reason, the next best value of C (C=0.3) was chosen for the model.

C Values	Validation Accuracy (%)
0.1	54,91
0.2	64,96
0.3	69,79
0.4	$\mathbf{73,12}$

Table 1: Performance comparison with different C values

4.2 Network

With that in mind, for the neural network, a variation of LeNet5 was chosen, which 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 [HQB20].

As each device implements the model and considering each having a different computational capacity, a small and simple network was selected in order for the computation to be the quickest possible but also having the capacity to gather information from the client's dataset.

4.3 Algorithm

The Federated Average algorithm (FedAvg) was picked for the sever aggregation part of the algorithm. The FedAvg consists on aggregating the weights of the clients according the number of images each client sees, with respect to the total number of images of the dataset. The FedAvg algorithm is simple, but useful enough in order to get an acceptable accuracy.

In addition, a variant of the Federated Average algorithm with server momentum was also considered, called FedAvgMomentum [HQB19]. Vanilla FedAvg updates the weights via:

$$w \leftarrow w - \Delta w$$

where

$$\Delta w$$

is the weighted average of the weight updates of the clients. To add momentum at the server side, a different computation is made instead

$$\nu \leftarrow \beta\nu + \Delta w$$

and update the model with:

$$w \leftarrow w - \nu$$

A comparison between the two algorithms was thought as interesting in order to see what impact the introduction of the momentum server had on the speed of convergence and oscillations, particularly in the scenario where clients have different data distributions.

5 Experiments

5.1 Federated Average

As baseline model, Federated Average (FedAvg) algorithm was chosen. In order to get an idea of the performance of the algorithm, a comparison with the centralised model was needed.

First of all, fine-tuning the parameters for the centralized model was necessary. A decision was made to adopt the number of epochs E = 100, as a trade off between performance and computational cost. The best learning rate the value LR = 2e-3 was discovered as resulting in better performance, because lower values had worse results, while higher values showed overfitting. With that in mind, in the federated model, the same learning rate was used, and a decision was made to use 20 rounds of 5 local epochs, to have a fair comparison between the two scenarios.

** In the plots from the federated experiments, the term epoch refers to rounds and not to local epochs. **

Figure 1 represents the difference in performance between centralised and federated model with different set configuration. It's evident and as awaited, that the centralised model has a better performance in all scenarios. The FedAvg does not lead to a good accuracy, hardly reaching 70%. Still, being a model without any further optimization, the accuracy was thought as acceptable.

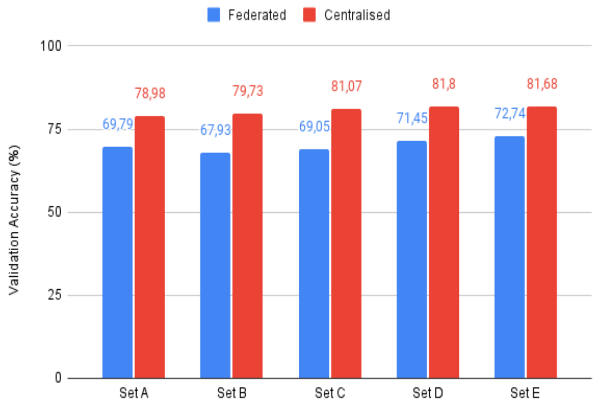

Figure 1: Accuracy comparison between centralised and federated model. (Set A: no normalization, Set B: batch normalization, Set C: group normalization (2-2), Set D: group normalization (2-4), Set C: group normalization (2-16)

5.2 Group and Batch Normalization

In order to be able to decide which model implementation is better, a comparison between normalization was made in table 2. What can be extracted from this experiment is that normalization is relevant for the accuracy. It's true that the accuracy decreases in the federated model by adding batch normalization (from 69,79 to 67,93 % for no normalization and batch normalization respectively). However, normalization seems crucial as adding group normalization added up to 4% in accuracy performance.

5.3 Different data sizes

In order to simulate different quantity of data that each client is going to have, a variable Delta was implemented in order to add randomness. The experiment was followed by changing the number of groups division of the group normalization layer 1 and 2. The results are represented in figure 2, illustrating with different group normalization parameters, the accuracy over the validation set.

Sticking with figure 2, over 8 tests done, there's no clear difference of a better performance with or

Model	Validation Accuracy (%)	Normalization
Federated	67,93	Batch
Federated	69,79	No
Federated	**73, 15**	Group (8-2)
Centralised	79,73	Batch
Centralised	78,98	No
Centralised	**81, 7**	Group

Table 2: Performance comparison with batch normalization for 10 rounds and 10 epochs

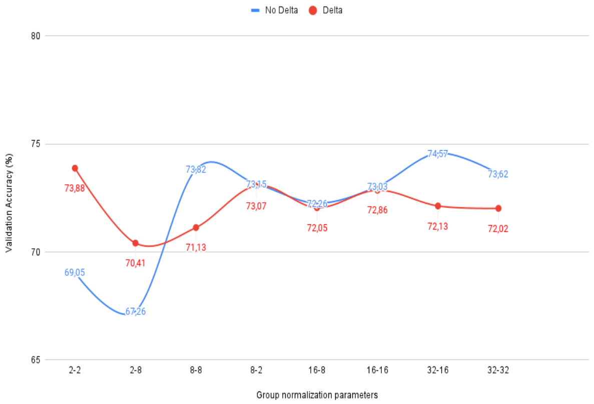

Figure 2: Accuracy over validation set with Delta = 100 variable activated or not

without Delta variable. Therefore, a new experiment was done, however this one just varying Delta to see the behaviour of the model, anything else remained unchanged. The results are presented in figure 3. As it can be seen from the graph, for a Delta varying from 50 to 300 the accuracy does not decrease, however, after Delta =400, it appears as the performance starts to decrease which was expected (from 73,44% to 70,64%), as the data among the clients start to differ substantially.

5.4 Dirichlet distribution

In order to achieve a better understanding of the possible accuracy outcomes when deploying the model,

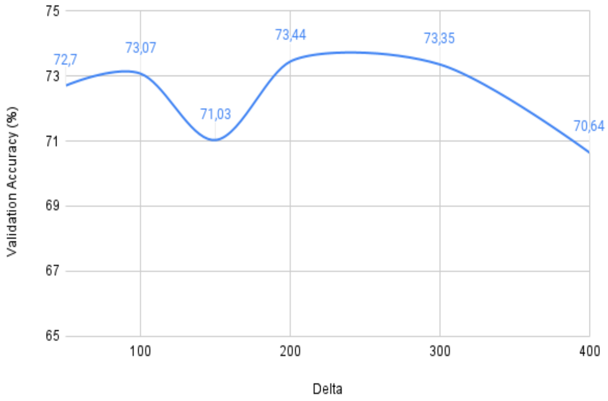

Figure 3: Accuracy over validation set with different Delta values variable

different data distribution between clients should be taken into account. In order to simulate client getting different data distribution, the Dirichlet distribution was used. The parameter Alpha, represents how different the distribution is between clients: with a greater Alpha, the distributions would be similar, with smaller Alpha, the distribution between clients differs. In particular, when Alpha is equal to 0 each client has access to data belonging to only one class.

In order to implement it, data already distributed with different Alpha parameters was taken from `https://github.com/google-research/google-research/tree/master/federated_vision_datasets` where Alpha goes from 0 to 100 for the CIFAR10 dataset.

The results are shown in figure 8. This image clearly illustrates the fact that expanding the difference in the dataset distribution between clients (lower Alpha), as expected, leads to the decrease in accuracy. Moreover, from the plots of training and validation accuracy (Figure 4, Figure 5, Figure 6), it can be seen how the smaller Alpha, the greater the number of oscillations. The accuracy on the training is higher because the smaller the Alpha, the smaller the number of classes each clients gets, hence a perfect prediction is expected.

To obtain results that are more similar to the IID

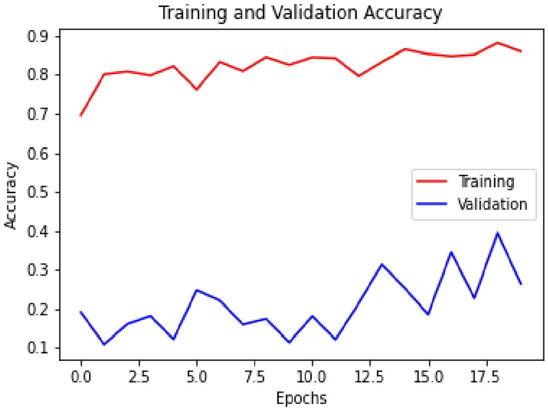

Figure 4: Accuracy using FedAvg with Alpha = 1

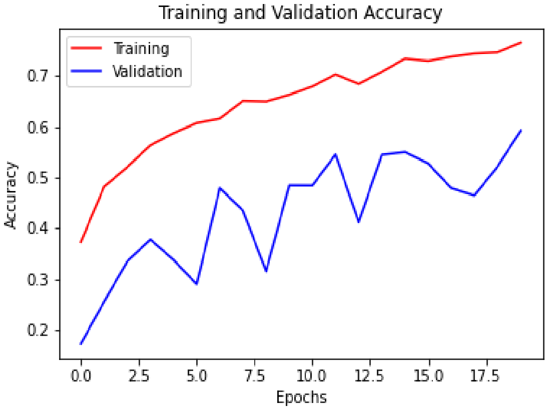

Figure 5: Accuracy using FedAvg with Alpha = 10

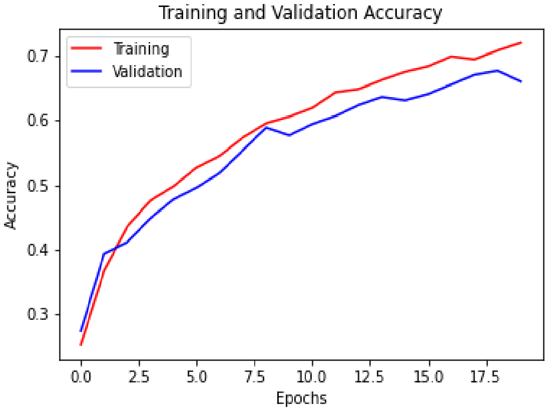

Figure 6: Accuracy using FedAvg with Alpha = 100

scenario also for lower values of Alpha, increasing a lot the number of communication rounds was needed. For example, in Figure 7 with Alpha = 1 validation accuracy reaches around 65% accuracy after 200 rounds like in the IID setting.

Finally, the figure 8 states that the group normalization model over-performs the model without normalization. This behavior goes along with the results gotten in the normalization experiments, and in this situation, normalization has an even more important role.

5.5 Federated Average Momentum (additional contribution)

The same setup of the previous experiments (Rounds = 20, Epochs = 5, Local Batch size = 32, K = 100, C = 0.3) is taken as starting point for the analysis. For the FedAvgM the same settings as in [REFERENCE FOR PAPER] was used: SGD as the optimizer for the server model, momentum (beta) = 0.9, and centralised model learning rate fixed to 1. From the accuracy and loss plots (9, 10, 11, 12) with IID data distributions for the clients, it can be seen how the results are very similar. However, the solution with server momentum has smaller oscillations and converges slightly faster.

An attempt was made to see if the behaviour

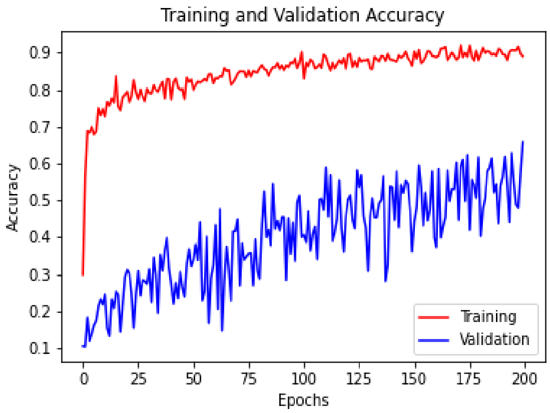

Figure 7: Accuracy with Alpha = 1

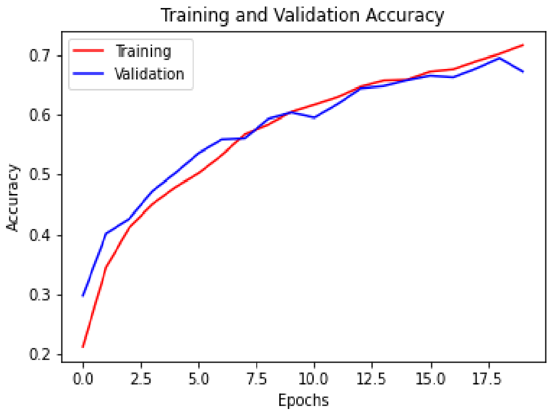

Figure 9: Accuracy using FedAvg with iid data distributions

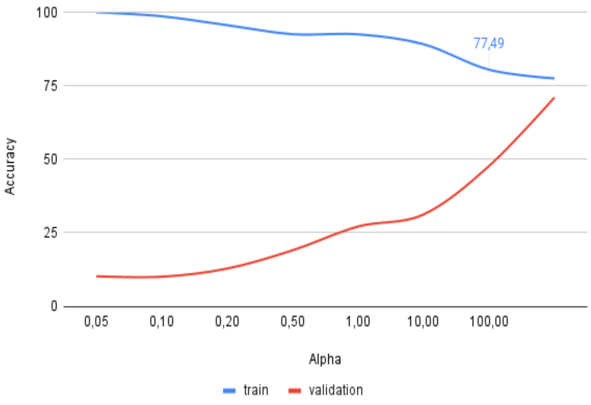

Figure 8: Accuracy over different values of Alpha

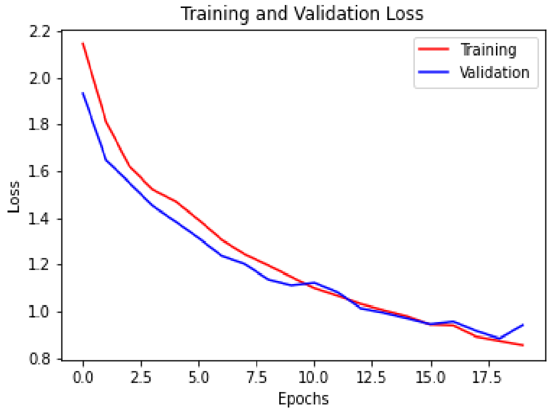

Figure 10: Loss using FedAvg with iid data distributions

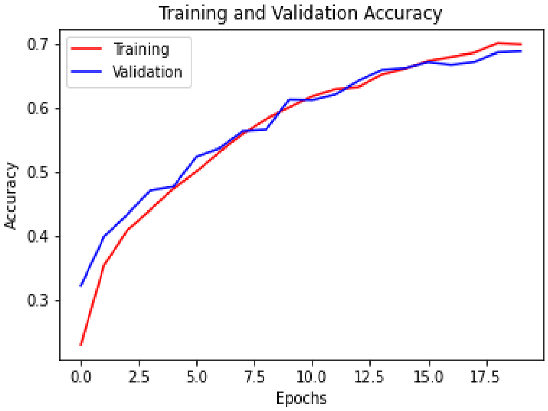

Figure 11: Accuracy using FedAvgM with iid data distributions

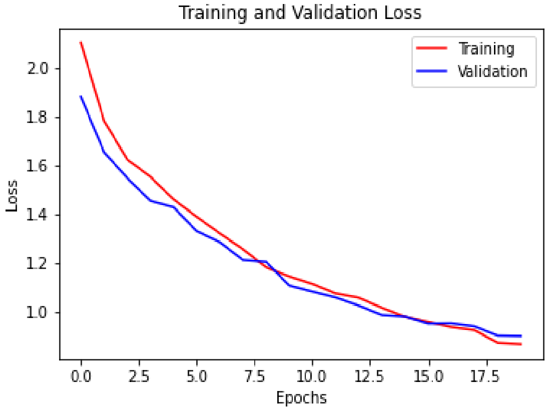

Figure 12: Loss using FedAvgM with iid data distributions

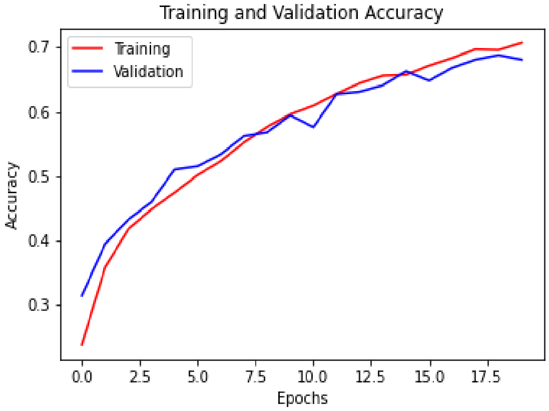

Figure 13: Accuracy using FedAvgM with Beta = 0.7

changed for different values of momentum . The plots of accuracies in figure 13 and in figure 14 show that when beta is lower the number of oscillations is higher, and their height is bigger.

With the non-IID data distributions for the clients, the results are slightly better than vanilla FedAvg's ones.

However, even in this case, there is a large presence of oscillations, particularly for small values of alpha, which makes comparison difficult. We can see that from figures 16 , 17 and 18.

Since in the original setting the number of rounds is too low to appreciate any major difference between the vanilla FedAvg and FedAvgM behaviors, a decision was taken to change the setup by increasing the number of rounds to 1000, and decreasing the number of local epochs to 1 so that the computation could still be handled by the system. The idea was to explore different regions where data distribution would differs, making reference to paper [Kon+16].

In the case with Alpha = 1 both the models reached accuracy around 0.7 and their trend is very similar, with wide oscillations in particular at the beginning.

Then another attempt was made to pick a case where the differences would be more evident changing Alpha to 0.05, and additionally changing the percentage of selected clients at each round (C) from 0.3 to 0.1. However, even on this occasion, as shown in

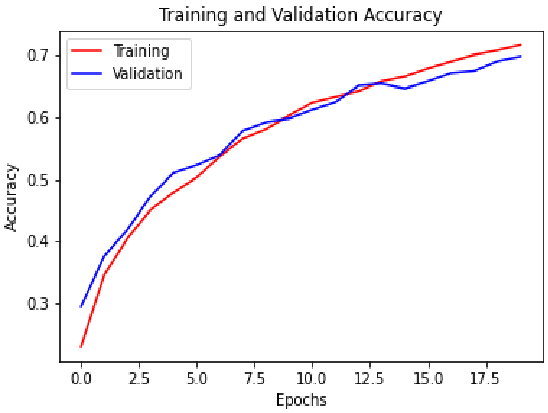

Figure 14: Accuracy using FedAvgM with Beta = 0.997

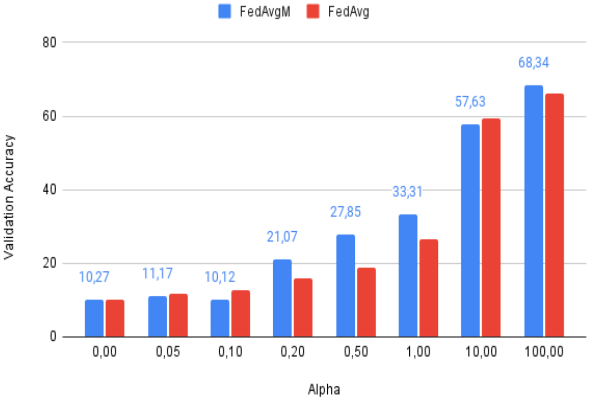

Figure 15: Accuracy comparison between FedAvg and FedAvgMomentum

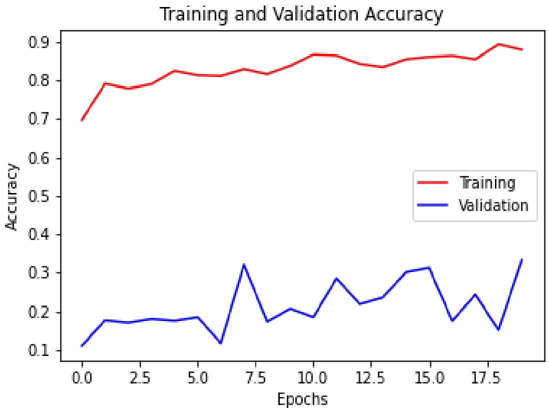

Figure 16: Accuracy using FedAvgM with Alpha = 1

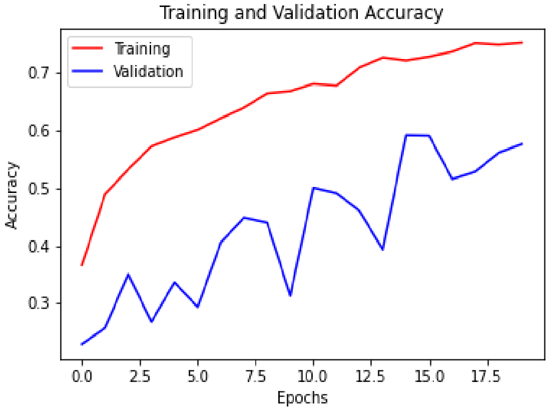

Figure 17: Accuracy using FedAvgM with Alpha = 10

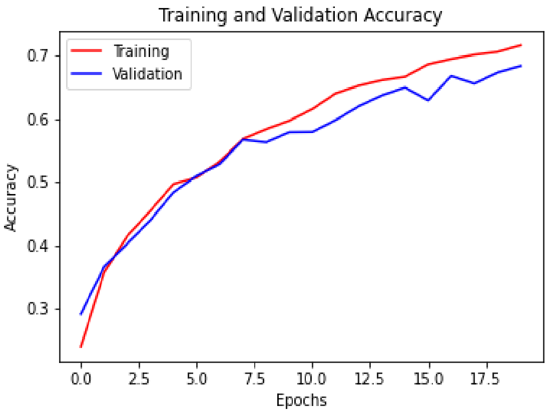

Figure 18: Accuracy using FedAvgM with Alpha = 100

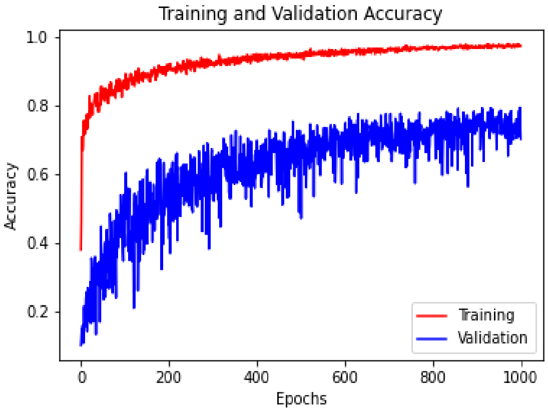

Figure 19: Accuracy using FedAvg with C = 0.3 and Alpha = 1

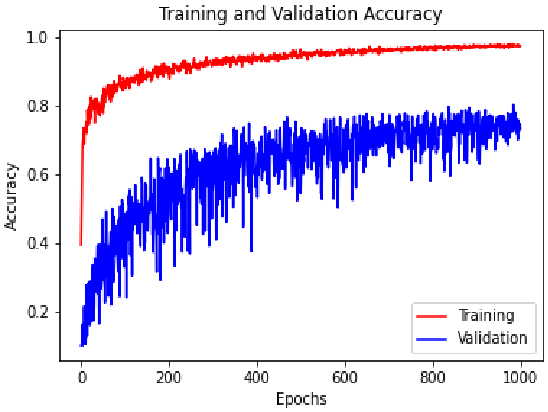

Figure 20: Accuracy using FedAvgM with C = 0.3 and Alpha = 1

figure 21 and in figure 22, the two algorithms achieve similar results. In this case FedAvgM's algorithm fails to achieve the expected accuracy. Probably with such low values of Alpha it would take an even greater number of rounds, which was not possible to test due to the limitations of the hardware at hand. The major difference between the two plots is at the beginning, where FedAvg starts with a higher value of training accuracy and converges faster.

5.6 Conclusion

The general idea that can be drawn from this work is that, although it doesn't have the accuracy of the centralised model, Federated Learning works and can be used on real life scenarios.

Moreover, as it has been shown with, the FedAvg does just fine if not fine tuned, and certainly normalization helps to improve the accuracy. As a general tendency, the normalization improves the accuracy, whether it is group or batch normalization (even though group normalization has better performances).

Additionally, the introduction of non IID and different quantity of data to the clients increased the difficulty of data treatment and learning. However, if the parameters didn't led to extreme cases, the ac-

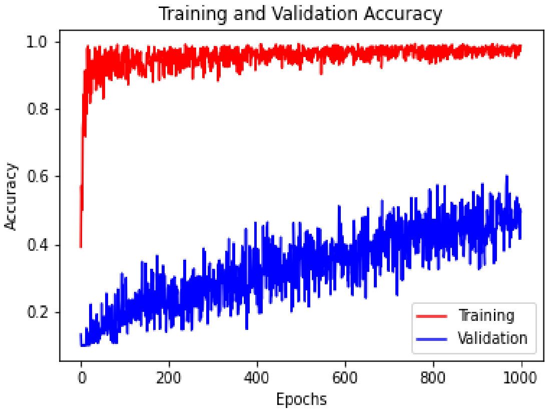

Figure 21: Accuracy using FedAvg with C = 0.1 and Alpha = 0.05

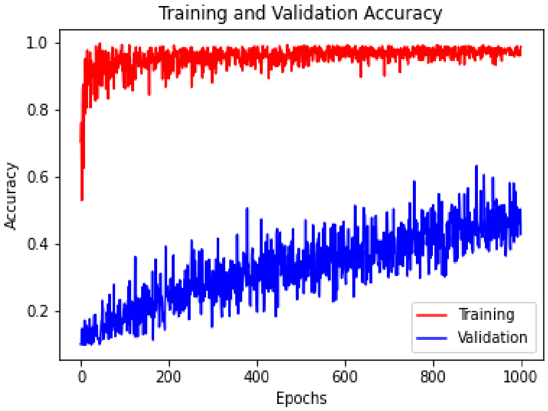

Figure 22: Accuracy using FedAvgM with C = 0.1 and Alpha = 0.05

curacy was still acceptable.

Finally, in order to deal with the non IID, the FedAvgMomentum, at first was thought as a feasible solution to improve the accuracy, however, the algorithm ended up not having the positive impact expected with the resources that were available.

References

[HQB19] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. "Measuring the Effects of Non-Identical Data Distribution for Federated Visual Classification". In: *CoRR* abs/1909.06335 (2019). arXiv: 1909.06335. URL: http://arxiv.org/abs/1909.06335.

[HQB20] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. "Federated Visual Classification with Real-World Data Distribution". In: *CoRR* abs/2003.08082 (2020). arXiv: 2003.08082. URL: https://arxiv.org/abs/2003.08082.

[Kon+16] Jakub Konečný et al. "Federated Learning: Strategies for Improving Communication Efficiency". In: *CoRR* abs/1610.05492 (2016). arXiv: 1610.05492. URL: http://arxiv.org/abs/1610.05492.