Federated learning: Collaborative data

Team members:
Christian Manuel Bartolomé Moreno
María Castellanos Ruiz
Marcos Espinosa Fernández
Pedro Herrero Herrero
Dennis Elizabeth Jara Díaz
José Fernando Viana Agudo

Coordinator team members:
Pablo González
Antón Makarov
Alexander Benítez Buenache
Maider Fernández Egido

Assistants team:
Adan Rodríguez Martínez
Claudio J. Salaroli

Acknowledgments

The team members are grateful with the GMV members -Pablo González, Antón Makarov, Alexander Benítez Buenache and Maider Fernández Egido - who provided numerous suggestions and helped us with technical questions along the Modelling Week.

This 14th Modelling Week would have been impossible without the effort and good management of Dr. Valeri Makarov. All team members recognize and thank his work.

Abstract

The main purpose of the Modelling Week is to promote the use of mathematical methods and models in research, industry, innovation, and management in the knowledge economy. Master students and participants work in small groups on real industrial problems proposed by companies under supervision of several qualified instructors. The 14th Modelling Week was organized within the Master in Mathematical Engineering program at the Faculty of Mathematics at the Complutense University of Madrid (UCM) in colaboration with the Institute of Interdisciplinary Mathematics (IMI). This special event was held in an online format from June 8 to June 12, 2020. [1]

The 14th Modelling Week's first problem was proposed by GMV company. The main aim of this problem is the analysis of the usability and efficiency of federated learning through a neural network model applied to the MNIST database using OpenMined PySyft framework. The scheme proposed to this work was: Create a model and train it with a complete, an incomplete and a biased MINST dataset; compare the result obtained in the different cases and discuss them; distribute the complete dataset in different workers and do federated learning; analize how well is performing this last model and try to improve it. This technical report explains the strategy adopted by the team members to solve this defiance problem and the obtained results.

Keywords: Federated learning, machine learning, private deep learning, neural networks, PyTorch, OpenMined PySyft, MNIST database.

Contents

1 Introduction

This Project has been developed with the goal of analyzing the usability and the efficiency of Federated Learning through a neural network model applied to the MNIST database.

Federated Learning is a simple yet, powerful way to train Deep Learning models. The most important feature to remark is that using this technique we bring the model to the training data, which allows whomever is creating the data to own the only permanent copy and maintain control over anyone that has access to it, improving privacy and ownership.

The basic data collection process takes images and transforms them into tensors. On a next step, these tensors are normalized and sent to the workers. This normalization consists in standardising our data. When a tensor is sent to the worker a pointer is returned to that tensor. Consequently, all the operations will be executed with this pointer that holds information about the data and then we send the model to the workers. With this process we are able to maintain the privacy.

It is important to note that, typically, Federated Learning pointers obtain the data from some external points, but for the purpose of this research we will make virtual workers to simulate them. The figure 1 shows some part of the process commented above.

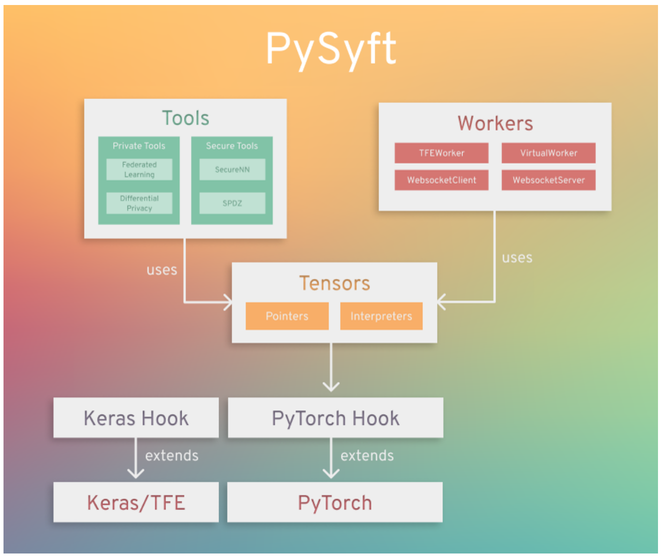


Figure 1: *Pysyft high level architecture taken from Pysyft Github web page [6]*

Let's mention two important applications of Federated Learning:

1. The first one is the ability of mobile phones to predict the next word users are going to write. This prediction should be made in a federated manner, because people don't want their private messages, which are the data used to train the model, to be sent to a central server. It is better to train it in each device.

2. Another application is related to medical data. Although some good models could be developed in order to predict the evolution of illnesses and could even save patients' lives, hospitals cannot share this kind of data to any external organisation, because people's privacy must be kept. Again, a good solution to this problem may be the use of Federated Learning models.

In this project, we have used MNIST database. It contains 70000 images of handwritten digits all in grayscale, with 28×28 pixels and centered. It is divided in 60000 images for training and 10000 for testing. (Figure 2)

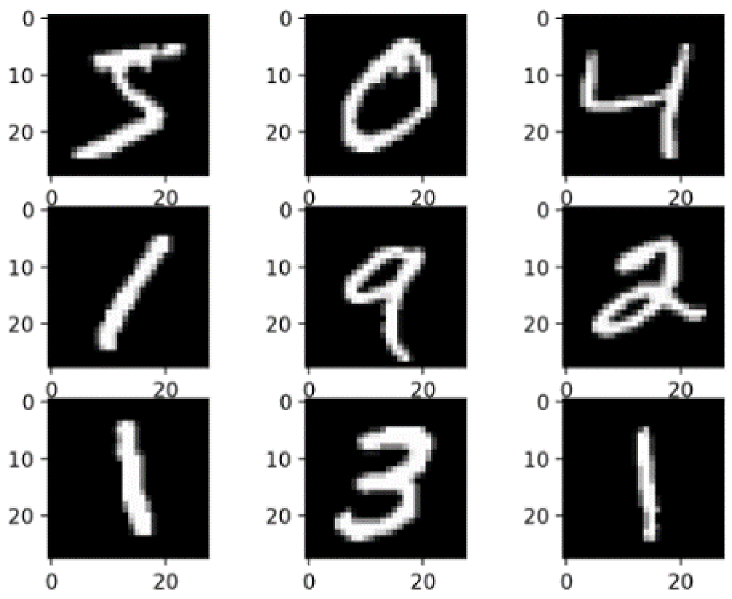


Figure 2: *Example of MNIST database images*

With the intention of defining our proposed Problem Scope, we started creating a model and training it with the suggested petitioned data. Afterward, comparing the results obtained in the different cases and discussing them. Continuing with a distribution of the complete dataset in different workers and applying federated learning. Lastly, seeing how well this last model performed and trying to improve it. To implement it we have designed a strategy divided in 3 phases:

- In first phase, we created a model without applying federated learning, using biased, incomplete and complete datasets.
- In second phase, we created a model applying federated learning using virtual workers with the complete and equally distributed dataset.
- And, in third phase, we created a model applying federated learning using workers with incomplete biased datasets.

In order to present the obtained results, we have used two different metrics: Accuracy and Balanced Accuracy. The first one is the result of the total correct predictions divided by the total results. On the other hand, Balanced Accuracy is calculated as the average of the proportion of correct predictions of each class individually. Additionally, we have used the Confusion Matrix, that allows us to visualize the actual values and the predicted values.

2 First phase

This phase was relevant because:

- It gave us a first approach to the Pytorch and Pysyft frameworks.
- It allowed us to understand how to prepare the different datasets.
- It gave us a framework to compare the performance of federated learning algorithms.

The workflow employed in the first phase is presented in figure 3. The work started with a literature review on the neural networks proposed for this problem, and got a set of convolutional neural networks. Next, we selected one neural network from the set of convolutional networks. Then we prepared three different datasets: One complete, another biased and finally another incomplete. We verified that the data partition was correct by using histograms over the training datasets. Then we applied the model and checked the accuracy. If the accuracy was under 97%, we selected another model and the model application process started again.

The neural network selected is based in [6] and [4]. We trained it with 3 epochs, a learning rate of 0.01, and a training moment of 0.5.

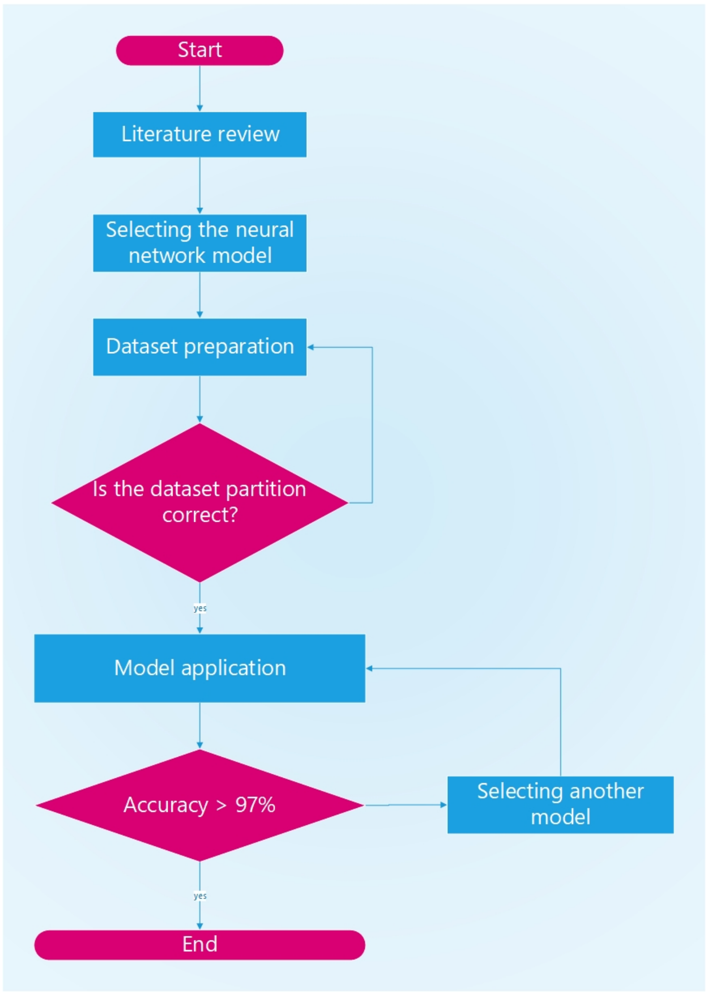


Figure 3: *Workflow for the first phase*

2.1 Analysis of complete dataset

The complete dataset was prepared with all digits as they came from the source. The histogram in figure 4 for the training data confirms that it is equally distributed, because each digit is around 6000 samples.

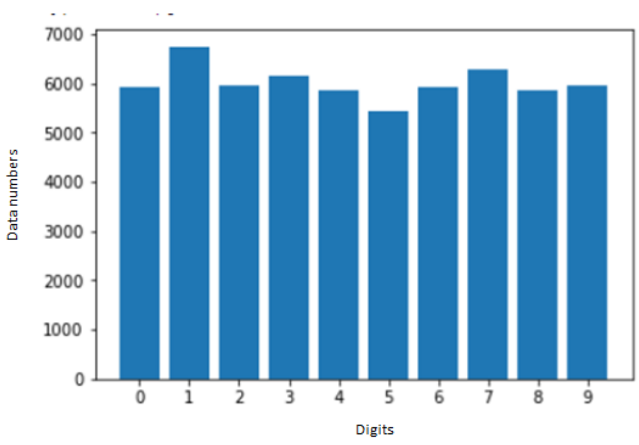
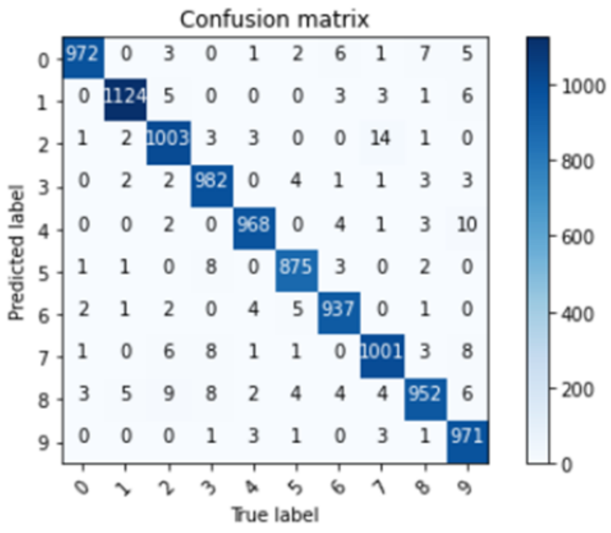


Figure 4: *Complete dataset partition verification*



	0	1	2	3	4	5	6	7	8	9
0	972	0	3	0	1	2	6	1	7	5
1	0	1124	5	0	0	0	3	3	1	6
2	1	2	1003	3	3	0	0	14	1	0
3	0	2	2	982	0	4	1	1	3	3
4	0	0	2	0	968	0	4	1	3	10
5	1	1	0	8	0	875	3	0	2	0
6	2	1	2	0	4	5	937	0	1	0
7	1	0	6	8	1	1	0	1001	3	8
8	3	5	9	8	2	4	4	4	952	6
9	0	0	0	1	3	1	0	3	1	971

Figure 5: *Confusion matrix of complete dataset*

The figure 5 shows the confusion matrix after we applied the model. The accuracy obtained was:

$$AC = \frac{9785}{10000} \approx 98\%$$

The main diagonal shows us that for this dataset the neural network obtained a large number of correct predictions for each digit.

2.2 Analysis of biased dataset

The biased dataset was prepared in a way that over 60% of data was the digit 0 and the rest was distributed between the remaining digits.

The histogram presented in figure 6 proves that the training dataset is significantly biased toward the digit 0.

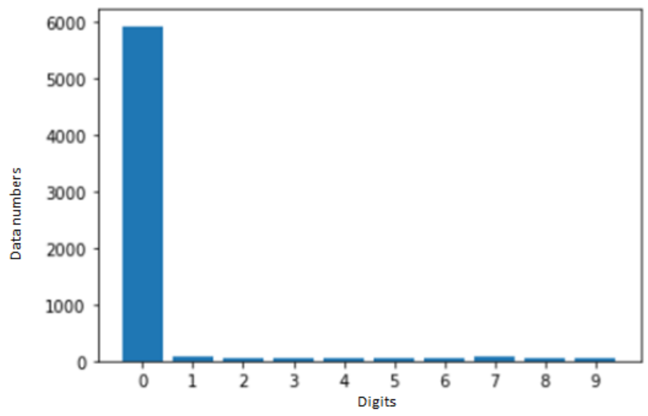


Figure 6: *Biased dataset partition verification*

Here the accuracy was 61%. On the one hand, the confusion matrix show us that prediction for digits $0, 1, 3$ and 9 had a large number of correct predictions but also a significant number of incorrect predictions (red rectangles in figure 7). On the other hand, the prediction for digit 7 had a large number of correct predictions and few incorrect predictions (blue rectangle in figure 7).

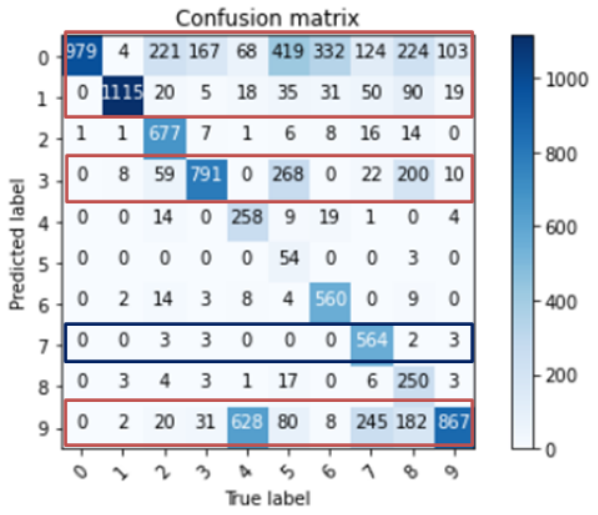


Figure 7: *Confusion matrix of biased dataset*

2.3 Analysis of incomplete dataset

The incomplete dataset was built only with 0, 1 and 2 digits. The figure 8 presents the histogram that verifies that the training data has only the digits 0, 1 and 2.

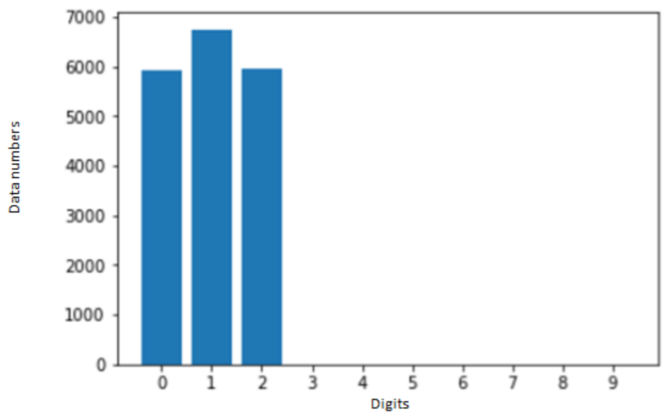


Figure 8: *Incomplete dataset partition verification*

The accuracy obtained here was 31% and the confusion matrix in figure 9 shows us correct predictions as well as a big number of incorrect predictions.

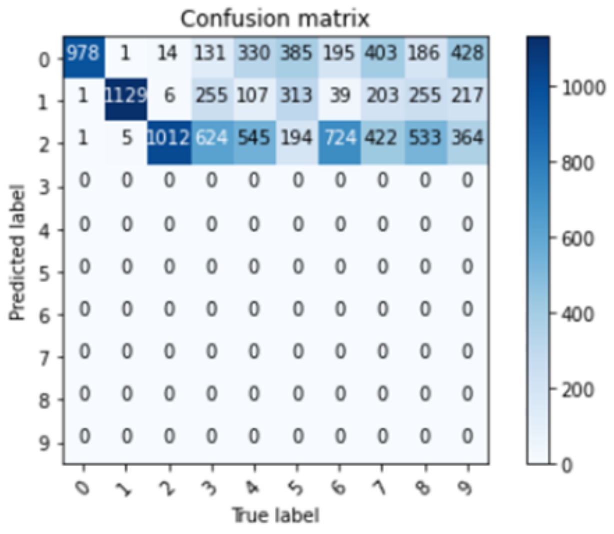


Figure 9: *Confusion matrix of incomplete dataset*

The main conclusion in this phase relies on the balanced accuracy plot showed in figure 10. It let us claim that the neural network makes better predictions when it is get supplied with a complete and equally distributed dataset.

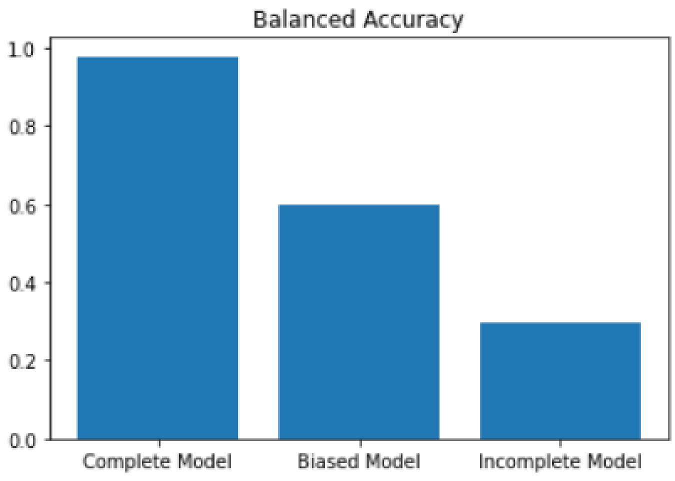


Figure 10: *Balanced accuracy plot*

3 Second phase

In this part of the project, we have developed, for the first time, a model in a federated manner. We have created three workers, called Marcos_Pedro, Chris_Maria and Fer_Liz and sent the third part of the data (20000 images out of 60000) to each one. Then, we have trained the model in each of the workers, one by one, as it is represented in figure 11. This is what we call an epoch.

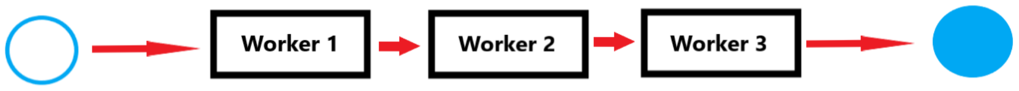


Figure 11: *Workers representation*

However, federated models are usually trained in several epochs, in order to improve the accuracy of the predictions, so the process of the previous diagram is repeated more times: the model is trained again using the data from each and every of the workers for the number of epochs that we want: for example, in this project, we have trained the federated model in three epochs.

We have used in this phase a database with equally distributed data, that is, we have more or less 10% of images of each of the digits from 0 to 9. When sending the data to the workers, we have kept the same distribution in all of them as in the complete database (figure 12):

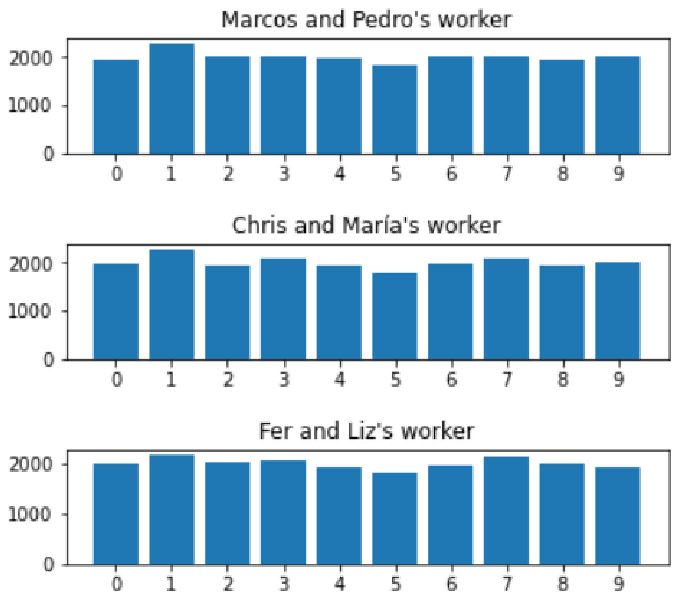


Figure 12: *Histograms for the datasets distributed in each worker*

In this particular case, in which data is uniformly distributed, the order of the workers does not matter, the results do not change if we alter their positions in the model. Nevertheless, if the distribution is not uniform, the order becomes important, results change when modifying it.

The results that we have achieved are:

- 93% of accuracy (9349/10000) in the first epoch.
- 95% of accuracy (9467/10000) in the second epoch.
- 96% of accuracy (9615/10000) in the third epoch.
- A balanced accuracy of 96%.

This result is slightly worse than which we have achieved in the 1st phase, 98%, and the reason is that here, the model uses just 20000 images in each of the three steps to train, whereas in the previous phase it used 60000, so the accuracy was expected not to be as high as in the not federated model.

The confusion matrix of this training is shown in figure 13. As we can see, the vast majority of the images of all the digits are correctly classified.

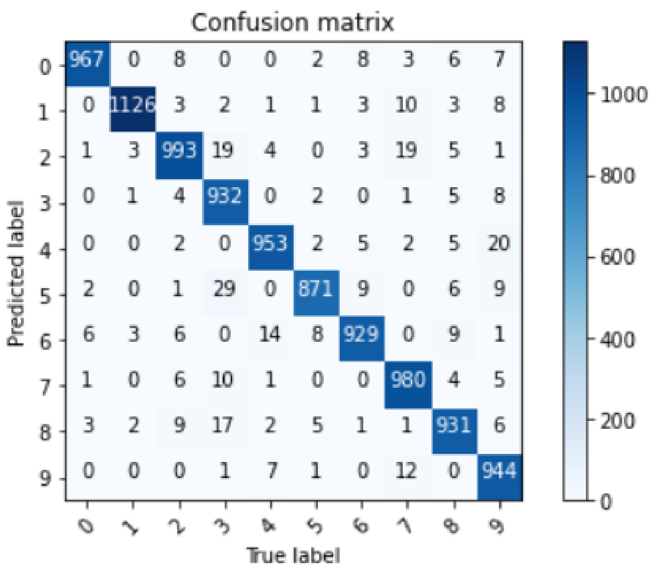


Figure 13: *Confusion matrix of second phase*

4 Third phase

This final section focuses on answering the next question: **How does Federated Learning work in some real life scenarios?**

First of all, I want us to imagine a toy real situation to facilitate the understanding of the experiment we executed: suppose we have 3 different hospitals where we want to study about some specific disease, like cancer. Everyone of these hospitals may specialize only in some types of cancer, so their specializations may all be different or they may have some of them in common. In any case, there is not doubt that, from a legal point of view, it is totally impossible to access patient's clinical histories. In other words, we are not able to do machine learning techniques because our models do not have the data the need to run. Here is the exact point where Federated Learning becomes a very useful and innovative tool. Let's see how it works with the next experiments:

4.1 Experiment 1

This experiment focus on what happens if the set of diseases each hospital treat are all different. Just in order to use our MNIST Dataset to illustrate what we are talking about, let's take a look at the next figure:

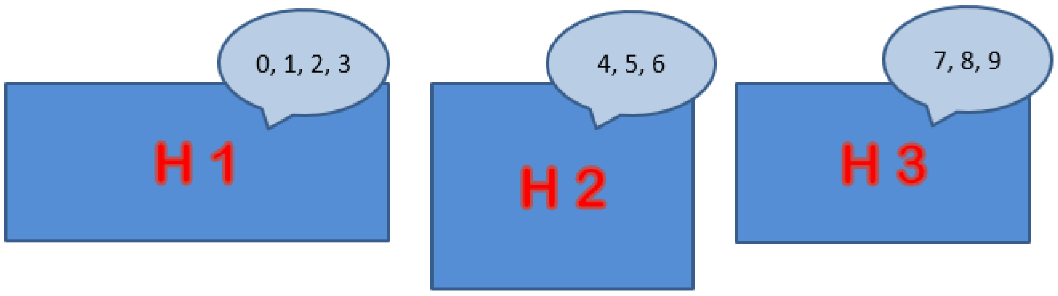


Figure 14: *Digits distributed per worker*

As we can see, figure 14 represents 3 different hospitals (everyone has its own dataset) which do not have digits in common. In this case, our datasets look like this:

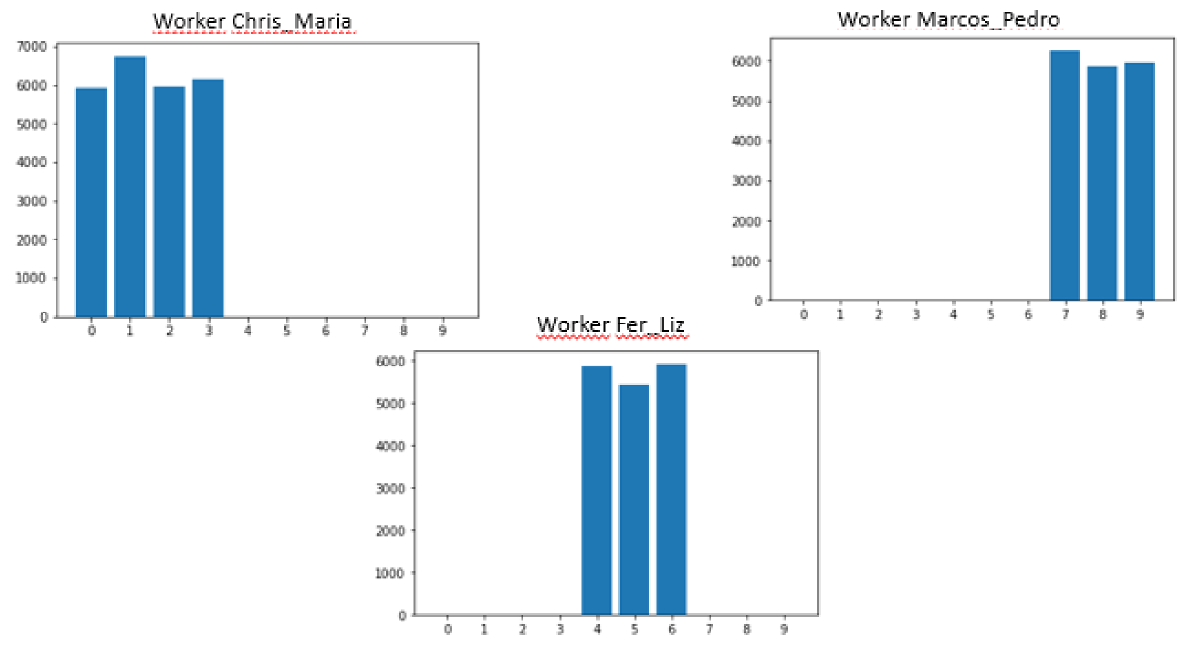
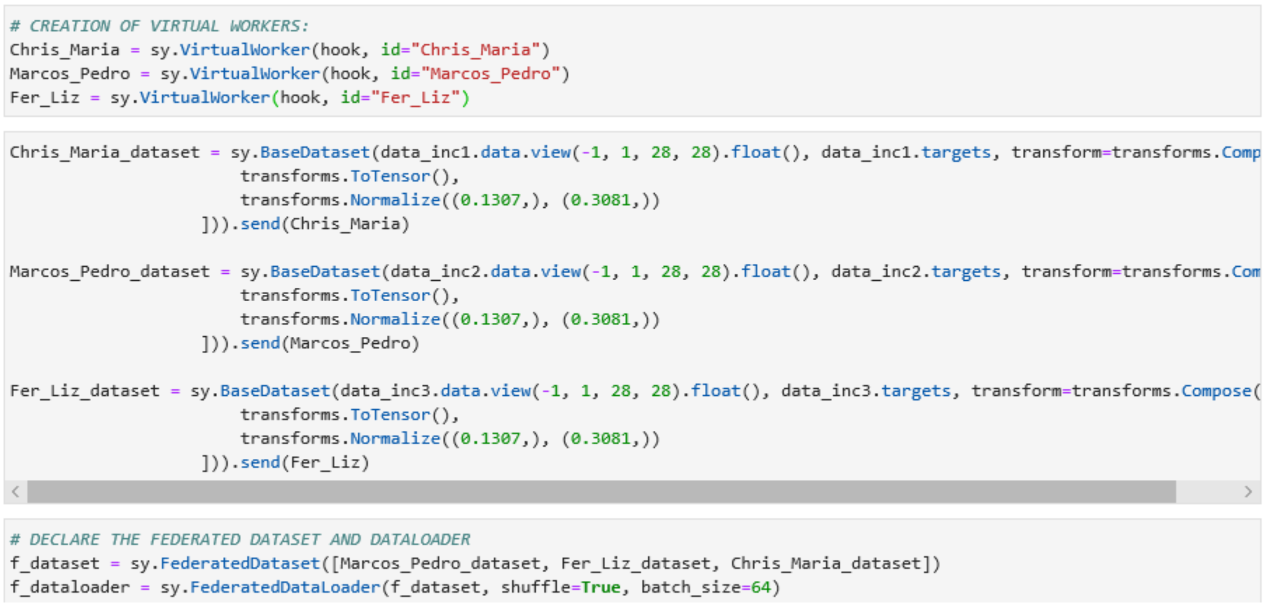


Figure 15: *First experiment: Histograms for the datasets distributed in each worker*

So, we have spread all of our 10 digits into 3 Virtual Workers without any kind of intersection between them. It's time to know the Federated Learning process:

Although we do not have any intention to fill this report with code pieces, we would like to present in this first experiment the process order we followed.



```
# CREATION OF VIRTUAL WORKERS:
Chris_Maria = sy.VirtualWorker(hook, id="Chris_Maria")
Marcos_Pedro = sy.VirtualWorker(hook, id="Marcos_Pedro")
Fer_Liz = sy.VirtualWorker(hook, id="Fer_Liz")
```

```
Chris_Maria_dataset = sy.BaseDataset(data_inc1.data.view(-1, 1, 28, 28).float(), data_inc1.targets, transform=transforms.Comp
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ])).send(Chris_Maria)

Marcos_Pedro_dataset = sy.BaseDataset(data_inc2.data.view(-1, 1, 28, 28).float(), data_inc2.targets, transform=transforms.Com
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ])).send(Marcos_Pedro)

Fer_Liz_dataset = sy.BaseDataset(data_inc3.data.view(-1, 1, 28, 28).float(), data_inc3.targets, transform=transforms.Compose(
                       transforms.ToTensor(),
                       transforms.Normalize((0.1307,), (0.3081,))
                   ])).send(Fer_Liz)
```

```
# DECLARE THE FEDERATED DATASET AND DATALOADER
f_dataset = sy.FederatedDataset([Marcos_Pedro_dataset, Fer_Liz_dataset, Chris_Maria_dataset])
f_dataloader = sy.FederatedDataLoader(f_dataset, shuffle=True, batch_size=64)
```

Figure 16: *Code fragment for the first experiment*

As we can observe, the first thing we did was to create the 3 Virtual Workers, which we have called with the name of the members of this group (note that these Virtual Workers are like Virtual Hospitals). Then, we spread the digits between them and it's very important to focus on the penultimate line of code. That line is where we chose the order our model was going to visit the Virtual Workers, so, in this example, our neural Work would first visit the Marcos_Pedro Virtual Worker, Fer_Liz after that and finally Chris_Maria. In other words, the model is firstly training with the 7, 8 and 9 digits, then with the 5, 6, 7 handwritten numbers and finally with the remaining ones.

Now, let's analyze the results of our model:

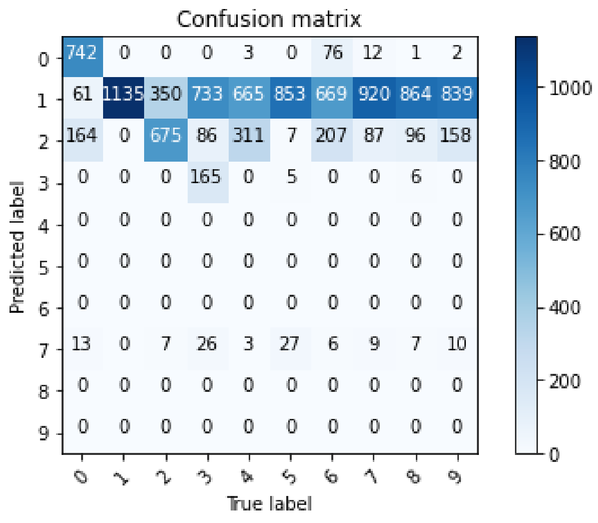


Figure 17: *First experiment: First confusion matrix*

- Number of Epochs: 1.
- Accuracy: 27%.
- Balanced accuracy: 25′83%
- Observations: The model tends to predict the digit 2 and it almost never predicts digits from the first and second Virtual Worker it visited. It's very good predicting digits 0, 1 and 3, but it fails because of that huge amount of 2 predictions.

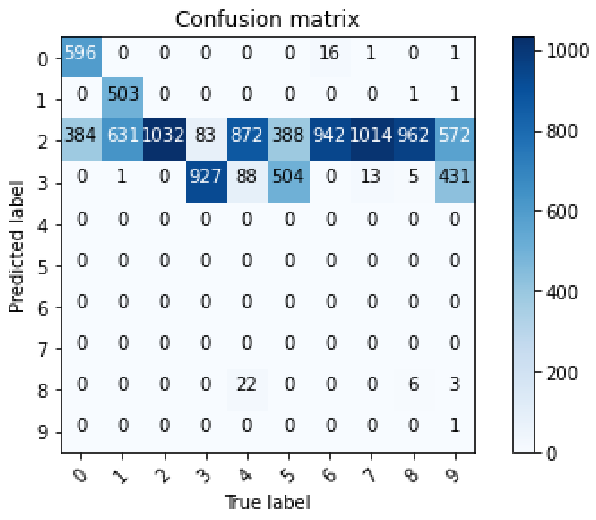
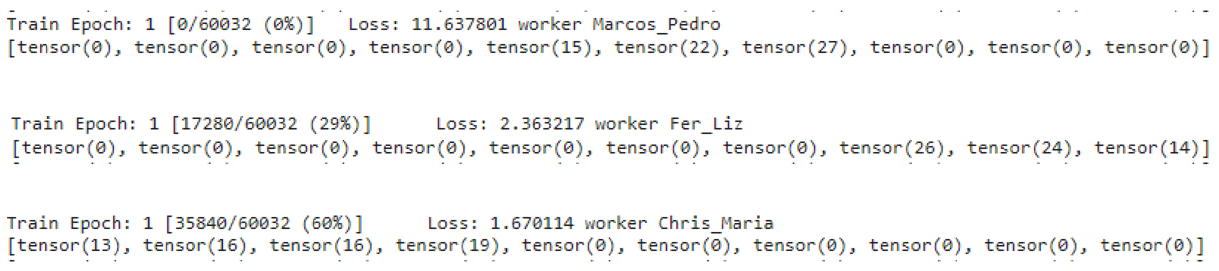


Figure 18: *First experiment: Second confusion matrix*

- Number of Epochs: 2.
- Accuracy: 31%.
- Balanced accuracy: 29′76%
- Observations: The model improves a little bit, but it keeps his tendency to predict the digit 3 and it's striking the way it almost never predicts digits from the first and second Virtual Worker it visited. It's very good predicting digits 0, and 1, but it fails because of that huge amount of 2 predictions. It also have problems to identify the digit 3, because it tends to confuse it with 5 and 9.

This results are very confusing, because we did not expect that debasing of predictions and how the model almost forgets the first visited Workers. So, we took a look into the way our model and process was getting data, paying special attention to how the model train. In order of that, we checked the batches spread, that is to say, the amount of data our model uses to train in every iteration (we used size 64), and we obtained the next results:



```
Train Epoch: 1 [0/60032 (0%)]   Loss: 11.637801 worker Marcos_Pedro
[tensor(0), tensor(0), tensor(0), tensor(0), tensor(15), tensor(22), tensor(27), tensor(0), tensor(0), tensor(0)]

Train Epoch: 1 [17280/60032 (29%)]      Loss: 2.363217 worker Fer_Liz
[tensor(0), tensor(0), tensor(0), tensor(0), tensor(0), tensor(0), tensor(0), tensor(26), tensor(24), tensor(14)]

Train Epoch: 1 [35840/60032 (60%)]      Loss: 1.670114 worker Chris_Maria
[tensor(13), tensor(16), tensor(16), tensor(19), tensor(0), tensor(0), tensor(0), tensor(0), tensor(0), tensor(0)]
```

Figure 19: *First experiment: Results*

Observations: The figure 19 shows the way our model trains into the 3 different Virtual Workers we are using.

- In the first photo, we can observe that in the first iterations (when the process has only ran 0%), the model trains with 15, 22 and 27 amounts of the digits 4, 5 and 6, respectively (note that amount adds up to 64).
- The second figure shows that, in the second training part, the model trains with 26, 24 and 14 amounts of the digits 7, 8 and 9, respectively.
- Finally, the last picture plots that the last part of the training occurs about 13, 16, 16 and 149 amounts of the digits 0, 1, 2 and 3, respectively.

This is totally correct and what we wanted to happen. So we did not found any failure in this checking and we just decided to see what happens if we run this experiment in a different way: what occurs if we switch the order of the Virtual Workers?

If we train with the 4 digits Virtual Worker in the first place, we obtain the next result:

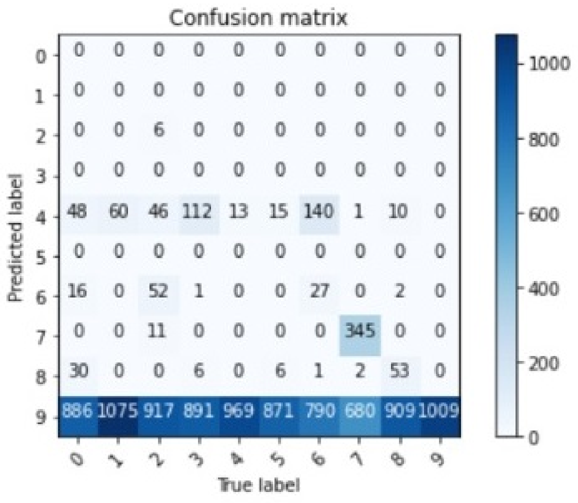


Figure 20: *First experiment: Third confusion matrix*

- Number of Epochs: 1.
- Accuracy: 15%.
- Balanced accuracy: $14'37\%$
- Observations: The model predictions are very bad as it has a big tendency to predict the digit 9 (it's actually the last digit of the last Virtual Worker it visited) and it's striking the way it almost never predicts digits from the non-last Virtual Worker it visited. Note that it also tends to predict some digit 4, as it's probably the closer digit to 9 in shape terms.

Henceforth, we will make some more experiments and try to find the reasons which explain this results.

4.2 Experiment 2

Now we wonder what happens if we have hospitals with different biased set of diseases. For instance, it may be the case that each hospital has at least one case of each type of cancer but they are not uniformly distributed and turn out some bias. Focusing now in our example, we distributed the complete dataset of digits in this biased way.

These graphics show workers distributions:

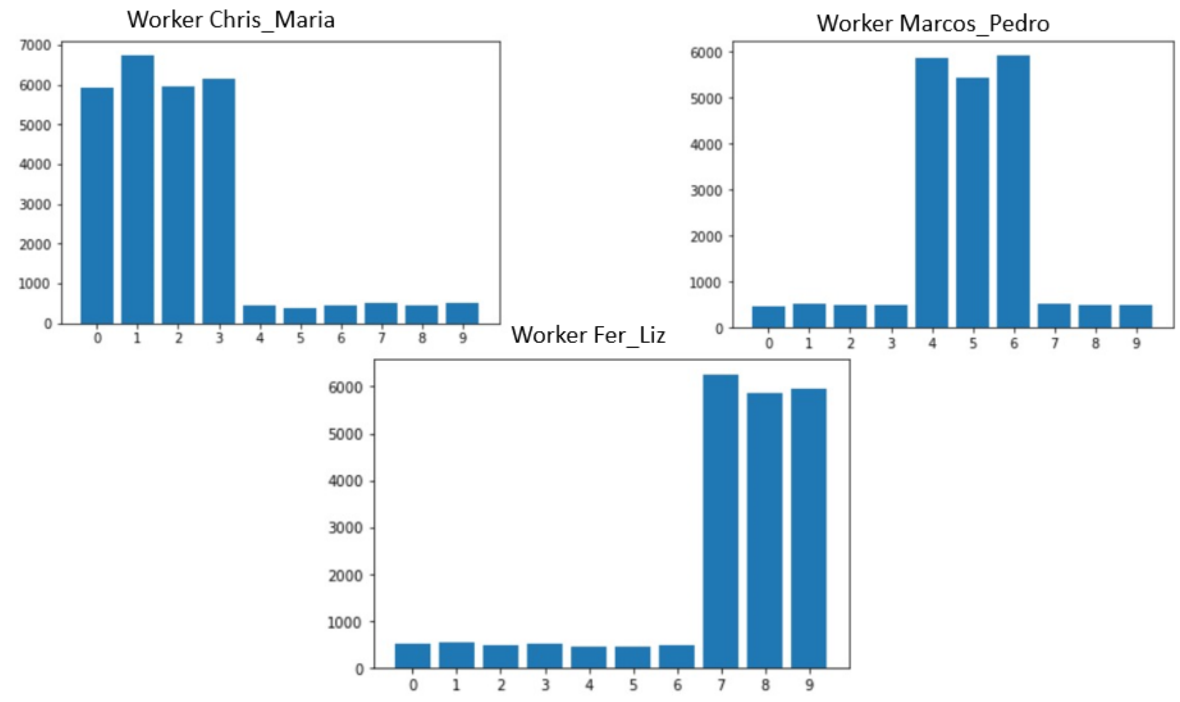


Figure 21: *Second experiment: Histograms for the datasets distributed in each worker*

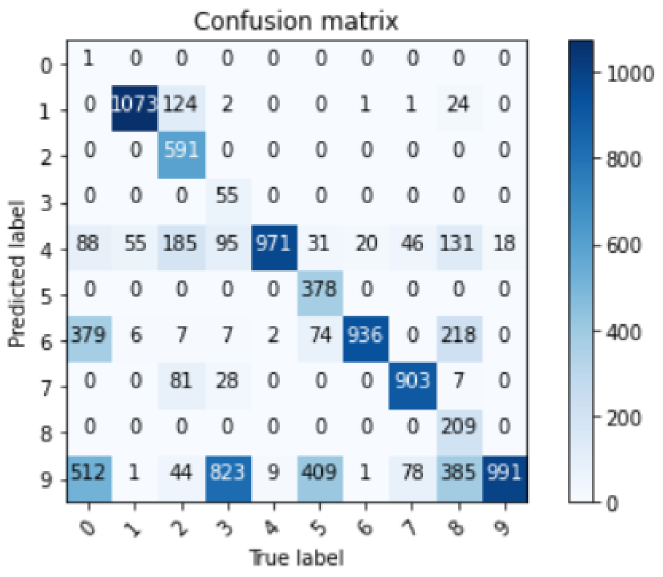
Note that, unlike the previous case, now we have all digits in each worker. Although the majority of numbers in each worker have a small frequency, it enables the improvement of the model accuracy because the weights of the neural network were updated with these numbers within the previous train. Then, when the model changes of worker, it remembers numbers with small frequency better than the incomplete model.

As in the other cases in this phase we have fitted two models only changing the order of workers. The first model was trained in the following order: the first worker is the one called Chris_Maria, which contains many zeros, ones, twos and threes, the second one is called Marcos_Pedro, which contains many fours, fives and sixes, and finally the worker named Fer_Liz, which contains many of the remaining digits.

Training the model for 3 epochs in this order we obtained the following metrics results:

- The model accuracy is 61%.

- The model balanced accuracy is 59.71%.



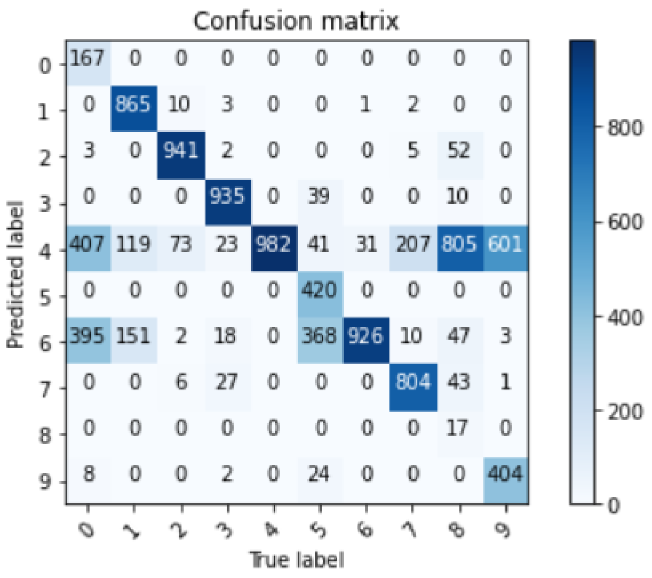
	0	1	2	3	4	5	6	7	8	9
0	1	0	0	0	0	0	0	0	0	0
1	0	1073	124	2	0	0	1	1	24	0
2	0	0	591	0	0	0	0	0	0	0
3	0	0	0	55	0	0	0	0	0	0
4	88	55	185	95	971	31	20	46	131	18
5	0	0	0	0	0	378	0	0	0	0
6	379	6	7	7	2	74	936	0	218	0
7	0	0	81	28	0	0	0	903	7	0
8	0	0	0	0	0	0	0	0	209	0
9	512	1	44	823	9	409	1	78	385	991

Figure 22: *Second experiment: First confusion matrix*

If we observe the confusion matrix, we can see the model predicts a lot of nines because nine is one of the latest numbers which the neural network trained the most. Moreover, this model also has problems predicting zeros and tends to predict the number four maybe because of its similar shape with nine or other reasons.

What occurs if we switch the order of the virtual workers in this experiment? Now, we train in the Chris_Maria virtual worker at the end. Training the model for 3 epochs in this new order we obtain the following metrics results:

- The model accuracy is 65%.
- The model balanced accuracy is 63.27%.



Predicted \ True	0	1	2	3	4	5	6	7	8	9
0	167	0	0	0	0	0	0	0	0	0
1	0	865	10	3	0	0	1	2	0	0
2	3	0	941	2	0	0	0	5	52	0
3	0	0	0	935	0	39	0	0	10	0
4	407	119	73	23	982	41	31	207	805	601
5	0	0	0	0	0	420	0	0	0	0
6	395	151	2	18	0	368	926	10	47	3
7	0	0	6	27	0	0	0	804	43	1
8	0	0	0	0	0	0	0	0	17	0
9	8	0	0	2	0	24	0	0	0	404

Figure 23: *Second experiment: Second confusion matrix*

If we observe the confusion matrix, the model trained with this new order also has a problem with numbers 0 and 4. Moreover, we can appreciate this model predicts worse numbers $7, 8$ and 9 because they are in the first worker instead of the last one like the model with the previous order. These differences suggest that order matters a little bit in this case in spite of their accuracy similarity.

4.3 Experiment 3

Now, what if we have a hospital with an equal distribution of the cancer cases and the others are specialized in some types?

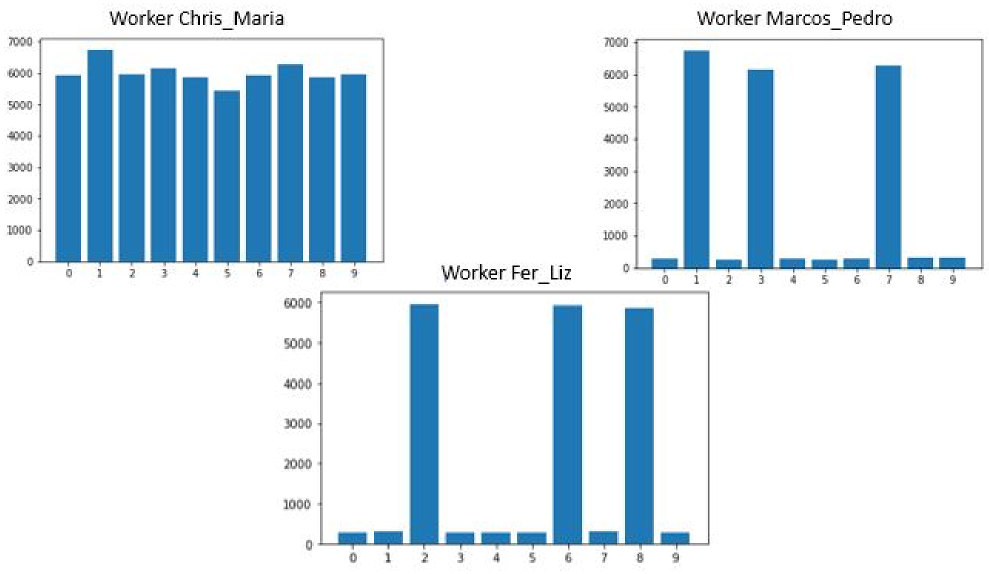


Figure 24: *Third experiment: Histograms for the datasets distributed in each worker*

Here we can see the distribution of the data in the workers. The Chris_Maria worker has a balanced level of cases and the other are biased: Marcos_Pedro towards $1, 3$ and 7 and Fer_Liz towards $2, 6, 8$. The order of the workers in this case is Marcos_Pedro, then Fer_Liz and finally Chris_Maria. This way we obtain an accuracy of 68%.

This confusion matrix shows a model that predicts quite good the numbers. However, there is a bias to the 2 and the 8 that were predominant in the second worker.

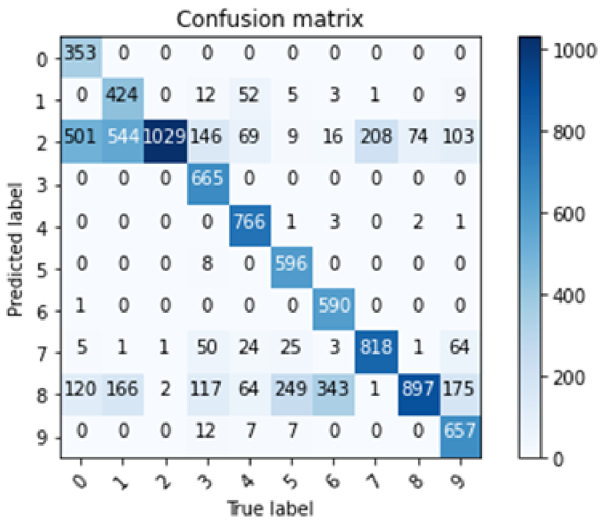


Figure 25: *Third experiment: First confusion matrix*

In order to observe if it is relevant the order of the workers with this structure of federated learning, now we try the order Chris_Maria, then Marcos_Pedro and Fer_Liz.

Using this new order of the workers we obtain less accuracy, a 55%. Looking at the confusion matrix we can see that the data is more biased towards the 6 that belonged to the final worker.

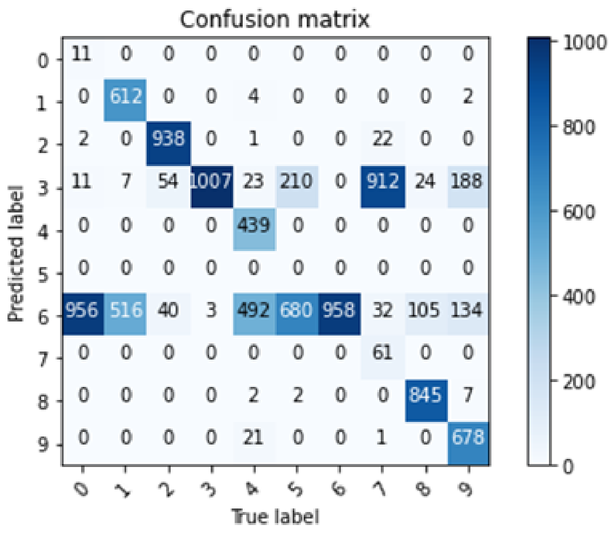


Figure 26: *Third experiment: Second confusion matrix*

5 Try to improve it

In this section, we tried to found a solution to face all problems we have find during this week. The most popular solution we found was what is called "Average Model" and what we illustrate in the next picture:

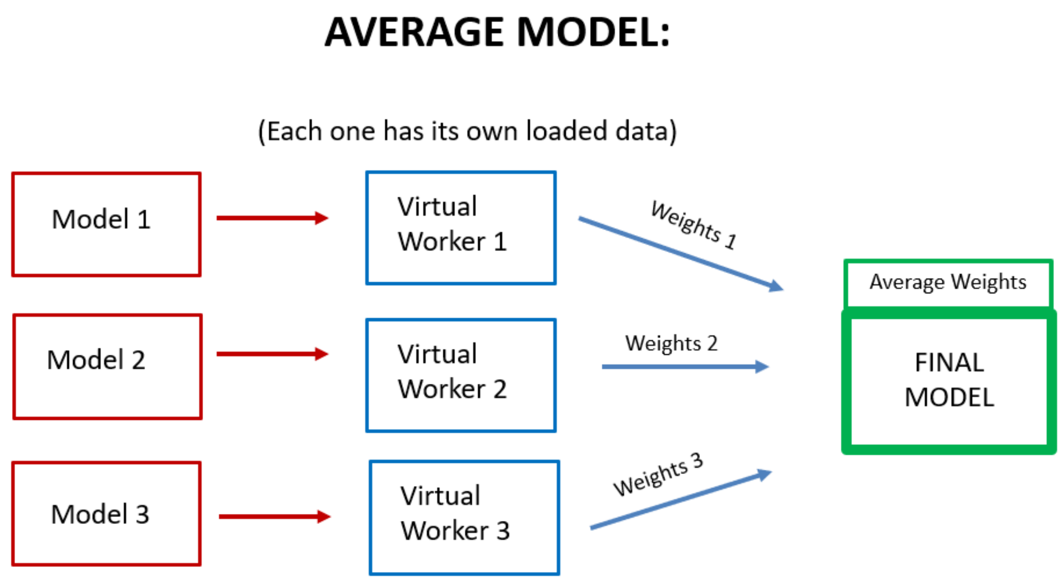


Figure 27: *The average model*

General idea: As we can see, this method consists in the union of 3 models which train in one and only Virtual Worker in an independent way. So, every model obtains his own weights according to the Virtual Worker digits it uses to train and all those weights are finally averaged in a final model.

6 Conclusions

Finally, the conclusions of the project are as follows:

- Not always more is better. We saw in phase 3 that adding virtual workers not so well distributed can make a poor performance.
- In terms of speed we realised that federated learning training takes almost double of time than the casual training.
- The importance of the model: as it might seem intuitive how good the model is strongly affects the results we got. We developed a neural network and we observed during the experiments along the week that there's room for improvement here.
- Instability: Related to the previous point, we experienced during the week instability with the model. The first analysis we took did not seem to enlighten where the problem was, so it will require a deep diving.
- The order matters: as we have observed, the model tends to boost the predictions of the digits which belong to the last Virtual Worker visited. In the other hand, digits from the first Workers are usually forgotten by the model. The fact which could explain this phenomenon is probably the dropout layers our neural network has, because they probably over-adjust the weights for the last VW.
- Privacy preserve: the final conclusion we would like to note is that all the Federated Learning process we have implemented can use tons of data without the necessity to know the information on them, what we think that will be very useful in the near future.

References

[1] Complutense University of Madrid (UCM). *XIV Modelling Week.* 2020. URL: http://www.mat.ucm.es/congresos/mweek/XIV_Modelling_Week/index.htm.

[2] V. Deepu, S. Madhvanath, and A. Ramakrishnan. "Principal component analysis for online handwritten character recognition". In: *Proceedings of the 17th International Conference on Pattern Recognition, 2004. ICPR 2004.* Vol. 2. IEEE, 2004, pp. 327–330. URL: http://ieeexplore.ieee.org/document/1334196/.

[3] M. Kang and D. Palmer-Brown. "A modal learning adaptive function neural network applied to handwritten digit recognition". In: *Information Sciences* 178.20 (Oct. 2008), pp. 3802–3812. URL: https://linkinghub.elsevier.com/retrieve/pii/S0020025508001564.

[4] Y. LeCun, C. Cortes, and C. J. Burges. *MNIST handwritten digit database, Yann LeCun, Corinna Cortes and Chris Burges.* 1998. URL: http://yann.lecun.com/exdb/mnist/.

[5] PyTorch organization. *Ecosystem | PyTorch.* URL: https://pytorch.org/ecosystem/.

[6] T. Ryffel et al. *GitHub - OpenMined/PySyft: A library for encrypted, privacy preserving machine learning.* 2020. URL: https://github.com/OpenMined/PySyft.

[7] Tensorflow. *Federated Learning | TensorFlow Federated.* URL: https://www.tensorflow.org/federated/federated_learning.

[8] Q. Yang et al. "Federated machine learning: Concept and applications". In: *ACM Transactions on Intelligent Systems and Technology* 10.2 (Jan. 2019).

[9] Q. Yang et al. *Federated Learning.* Vol. 13. 3. Morgan & Claypool, 2020, pp. 1–207.

[10] N. Yu, P. Jiao, and Y. Zheng. "Handwritten digits recognition base on improved LeNet5". In: *Proceedings of the 2015 27th Chinese Control and Decision Conference, CCDC 2015.* Institute of Electrical and Electronics Engineers Inc., July 2015, pp. 4871–4875.