UNIVERSIDAD
COMPLUTENSE
MADRID

gmv
INNOVATING SOLUTIONS

Federated learning:

Does everyone play fair?

Team members:
Marta Centellas Nadal
Marina Girón Lafuente
Andrea Gutiérrez Mato
Alicia Morales López
María Segura Miñano

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

Acknowledgments

We would like to thank the members of GMV - Pablo González, Antón Makarov, Alexander Benítez and Maider Fernández - for their constant help and interest, and for solving every doubt we had.

We would also like to thank Valeri Makarov, director of the XV edition of the Modelling Week. All team members recognize and thank this work.

Abstract

This work has been developed during the XV Modeling Week of the Complutense University of Madrid. The aim of this week is to promote the use of mathematical methods and models and their application in real life. The models range from those studied in the Master's Degree in Mathematical Engineering to new and innovative ones, as is the case of Federated Learning, which is discussed in the development of this work.

During this week, the Master students work in groups of about five people on real industrial problems proposed by companies under supervision of several qualified instructors. The 15th Modeling Week was organized once again as part of the program of the Master's Degree in Mathematical Engineering of the Faculty of Mathematics of the Complutense University of Madrid (UCM) in collaboration with the Institute of Interdisciplinary Mathematics (Instituto de Matemática Institute of Interdisciplinary Mathematics (IMI). This event was held for the second consecutive year in online format and lasted one week, starting on June 7, 2021 with the exhibition of the problems and ending on June 11, 2021 with the public defense of the results obtained by each group, [1].

This paper analyzes the problem proposed by GMV for this 15th edition. As a solution to data privacy issues in Machine Learning, Federated Learning arises. Federated Learning is a powerful way to train Deep Learning models in which, instead of bringing training data to the model (as usual Machine Learning models do), brings the model to the training data. In this work we aim to study the idea and main concepts of Federated Learning. For this purpose, we will train a global model using this method for digit classification when the data are in three different nodes and are restricted. To observe the advantages and disadvantages that this type of collaborative models presents, we are going to compare the results obtained with the results of training models with restricted data from each of the nodes (non-collaborative models) and also with a usual Machine Learning Model with all data on a local server. Finally, we will see what happens when one of the nodes has been attacked and its data is corrupted.

Keywords: Attack, Convolutional Neural Network (CNN), Fast Gradien Sign Attack (FGSA), Federated Learning, Machine Learning, MNIST, Nodes, Training.

Contents

List of Figures

Chapter 1

Introduction

Standard Machine Learning requires the centralization of training data on some kind of central server. However, this can be a problem at times. For example, suppose we want to implement a predictive model for nationwide breast cancer. Due to privacy issues, hospitals cannot give us their data. Consequently, we would only be able to create a model trained on a few thousand patients from a single hospital, and, without access to sufficient data, the model will not be able to reach its full potential and, ultimately, will not be able to make the transition from research to clinical practice.

In order to provide answers to this type of problem, Federated Learning arises. Federated Learning is a Machine Learning method that does not require the transfer of confidential information: instead of sending the data to a central server, where the model is trained, the algorithm is trained in parts in each place where the data is located and then combines everything learned in a single model. In this way, the data is never exchanged; the only thing that is transferred is the algorithms, which cannot be designed in reverse to reveal that data.

However, since in Federated Learning the training data is not stored in a centralized server, i.e., we do not have access to the data, it could happen that a malicious user attacks our model, for example, altering the data used to train the model, or altering the model itself, with the aim of compromising the global model. So, are we safe in Federated Learning against attacks from malicious users?

In this work we aim to familiarize ourselves with Federated Learning, studying the advantages and disadvantages it presents, as well as studying some specific types of attacks. For this, we take the MNIST dataset, which consists of handwritten digits from 0 to 9, and we will develop the following steps:

- We start by implementing a convolutional neural network (CNN) for digit classification using centralized Machine Learning algorithms, i.e., with the original dataset.
- Then, with the data of the original dataset distributed in three parts (nodes), we will make three Machine Learning (ML) models in which each one will be trained with the data of one of the parts without having access to the data. In this way we will have three models with restricted data.

- We will train a global model using Federeted Learning, so that each part does not disclose its own data. The results obtained will be compared with the previous results (trained by each part with its own data).
- Finally, we will train our Federated Model with poisoned data and analyze the results. We will also study other types of attacks.

Before starting with the elaboration of these models, we define the data set we will work with, as well as the neural network architecture we will use for the elaboration of all of them.

We are going to work on a dataset called MNIST [3] which consists in images of digits from 0 to 9 with their respective labels (Figure 1.1). This dataset has 70,000 images in total.

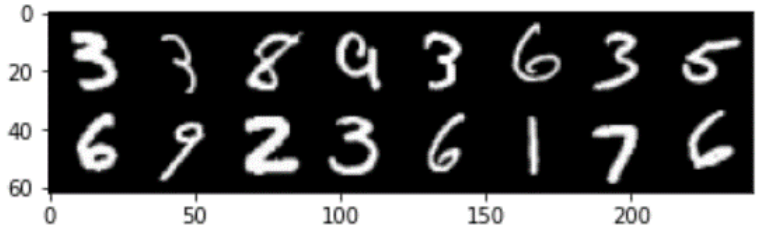

Figure 1.1: Sample dataset. The labels of this images are 3 3 8 9 3 6 3 5 6 9 2 3 6 1 7 6, respectively.

To build the model that we will use throughout the work, we follow the next steps:

- First, we define a Convolutional Neural Network (CNN) as in Figure 1.2.
- Then, we need to define a loss function and a optimizer. We use a Classification Cross-Entropy loss and SGD with momentum.
- We choose other parameters of the model:
  - The batch size, that is, the number of samples that will be propagated through the network. In our case we fix batch size = 16.
  - Epochs, that is, the number of times we want to train the net. For now, we choose epochs = 2.

In order to train our network and subsequently evaluate the model obtained, we separated the initial dataset into training set and test set. The training set has 60,000 images while the test set has the other 10,000 images.

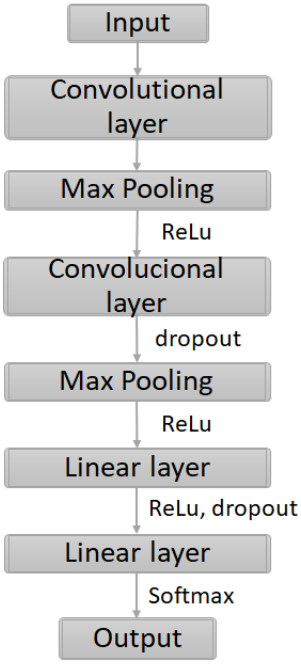

Figure 1.2: Convolutional neural network architecture scheme.

All work will be done in Python's Pytorch and PySyft environment [4], [5].

Chapter 2

Centralized model

In this first chapter we are going to train the network for the classification of digit images using these data under the assumption that we have all of them centralized, i.e., we have full access to the dataset MNIST. We train our network using the training set. Remember that we put two epochs. Evaluating the model with the test set we obtained an accuracy of 90%. This means that it predicts correctly most of the cases. It is quite important to know in which digits the neural network encounters more difficulties, so we are going to obtain the exact accuracy of each class in Figure 2.1.

Class	Accuracy
0	95.9
1	96.1
2	93.4
3	91.3
4	89.4
5	81.1
6	90.9
7	88.5
8	82.2
9	87.1

Figure 2.1: Accuracy by classes.

It can be easily observed that the digits that have been worst classified have been fives and eights. In order to see with which numbers the model has confused them, we construct the confusion matrix (Figure 2.2).

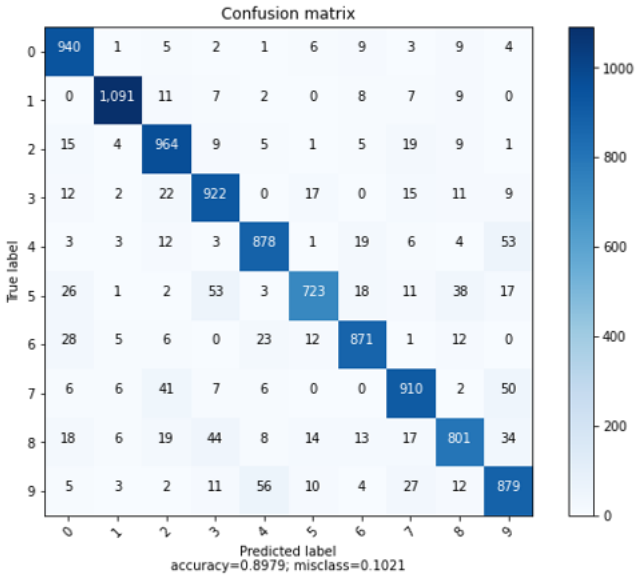

Figure 2.2: Confusion matrix of the CNN with two epochs. Centralized model.

With this matrix, we can refine the obtained accuracies and get more interesting information. It is clear that many nines are classified as fours, 56 to be exact, and that reciprocally 53 fours are classified as nines. In addition, fives are confused with threes and with eights. These types of errors makes sense because the shape of the digits in question are similar. So we can say that the network has learned correctly.

If instead of training the network with 2 epochs, we use 4 epochs, we obtain an accuracy of 93%, and with 6 epochs we also obtain an accuracy of 93 %. Therefore, it seems that we cannot improve the efficiency of our network by increasing the epochs from 4. But since our goal is not to build the most efficient neural network, but to understand the difference between centralized and decentralized models, we will leave the network we have explained with two epochs, unless otherwise stated.

In addition, we can see that the confusion matrix obtained by taking four epochs is very similar to that of two epochs, but as the precision increases, the number of failures obviously decreases, we see for example a notable decrease in the 9 classified as 4, as well as in the 8 classified as 3.

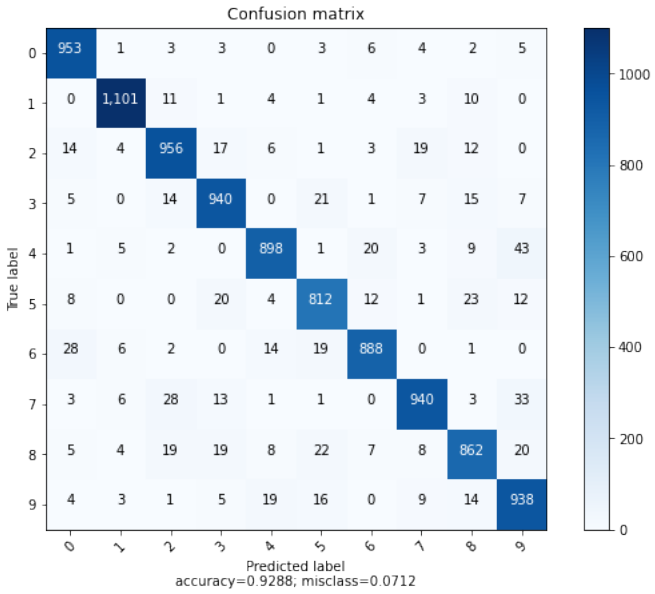

Figure 2.3: Confusion matrix of the CNN with four epochs. Centralized model.

In summary, in this first chapter we have created a CNN for digit classification under the assumption that we have full access to the data and that they are centralized and we have obtained an accuracy of 90%.

Chapter 3

Federated model

Federated Learning is a Machine Learning technique that aims to build systems that learn on decentralized data, that is, datas that are not on the same machine. In this way the data remains in the hands of its owner, which helps to improve privacy conditions, and at the same time the model is trained with data from different machines which helps to improve its accuracy. A clear example of privacy is mobile phones. Users are interested in matching ads to their needs or having the predictive keypad work for them, however, they do not want their data to be publicly owned. Federated models are trained on our data but without actually owning it.

Other example where there is a clear interest from different entities in having a good model is in the case of medicine, as previously mentioned. For example, a model that predicts whether a person will suffer from hypertension, a very common disease that causes a lot of damage before it is detected and whose early detection would save many lives, would be very useful. However, if we only train with data from our hospital, the information is reduced and we will not predict well enough. As we obviously cannot use data from other hospitals because of patient privacy, using a federated model would allow us to generate a more accurate model.

Given the current awareness with data privacy, as well as the creation of new protection laws Federated Learning is booming and will probably continue to do so going forward. In fact, large companies such as Apple and Google have already invested heavily in this technology.

We can differentiate between two types of Federated Learning:

1. Train independent models at each node with the available local data and send these trained models to a central server. In this server, a weighting of the network weights is performed to obtain a master model that has been learned with all the training data.

2. Train a model from worker to worker in order. A worker trains with its local data and this model is retrieved for another worker to train. This step is performed for all workers as many times as epochs have been defined. In this way, the final model will have passed through all the workers being trained little by little by each one of them.

In this section we are going to create the same model as in the local case, that is a convolutional network but this time without the data being available to us and therefore we will not be able to extract from them certain information of interest, such as whether the data are balanced or well labeled. Specifically, we will work on three modules.

We will separate this chapter into two parts, as we have done in the development of the week. We will begin by training the neural network with data from a single node and then we will move on to training it with data from all of them, we will see how. When training only with data from one node, we will have less information available, therefore, we expect to obtain less accuracy. This is one of the advantages of federated, we have much more information and therefore we obtain much more accurate models.

3.1 Single node training

The first step in the development of this type of model is to work with data from a single, but non-local, source. We have worked on three different nodes, which contain one third of the training data from the MNIST database. The objective is to train the network with this data and compare it with the local result. Then, we will train the network with the data from all the nodes and compare the results as well.

Therefore, we first train on each node with the available data and then test each node with the same test set as in the local case. We obtain the results shown in the Figure3.1.

	Node 1	Node 2	Node 3
Accuracy	82 %	83%	82%

Figure 3.1: Accuracy obtained for each node

In addition, we have checked the accuracy obtained in each of the classes, so we have been able to observe which are the worst classified by the neural network. We can see the results obtained in the Figure 3.2

Class	Node 1	Node 2	Node 3
0	92.9	93.6	93.3
1	95.0	92.4	93.2
2	83.3	79.5	80.8
3	78.2	86.1	81.8
4	74.1	82.2	81.7
5	80.8	77.9	75.0
6	90.6	87.6	87.7
7	85.0	84.6	78.0
8	72.5	71.6	70.6
9	73.5	75.7	77.2

Figure 3.2: Accuracy obtained for each class at each node

Regarding the accuracy of each class we can see that the results are quite stable. For example, in the case of numbers 8 and 9, in all nodes the accuracy is around 70% or 75%, respectively. This percentage is lower than in all other classes. On the other hand, the class of 0 which in all of them is around 93% and is the highest of all. Moreover, the accuracies obtained are similar to those of the local model.

Actually, these accuracies should not be directly compared with those of the local model, since the local model has been trained with three times as much data. A fairer way to compare is to change the training conditions, if instead of using two epochs we use six, that is three times as many, we could compensate for having one third of the data. After making this comparison, the results obtained were as follows:

	Acuracy			
Epoch	Node 1	Node 2	Node 3	Local
2	82 %	83%	82%	90%
4	88%	87%	83%	93%
6	89%	90%	90%	93%

Figure 3.3: Accuracy obtained according to the number of epochs considered

As we can see, the accuracies that are comparable have been surrounded in red, since we have one third of the data and three times the number of epochs, these accuracies are very similar, as we would expect. As already mentioned, increasing the local epochs also improves but stagnates at 93%, which indicates what will be the highest accuracy that we could expect with this network and these data.

Finally, let's see what is the confusion matrix obtained at each node:

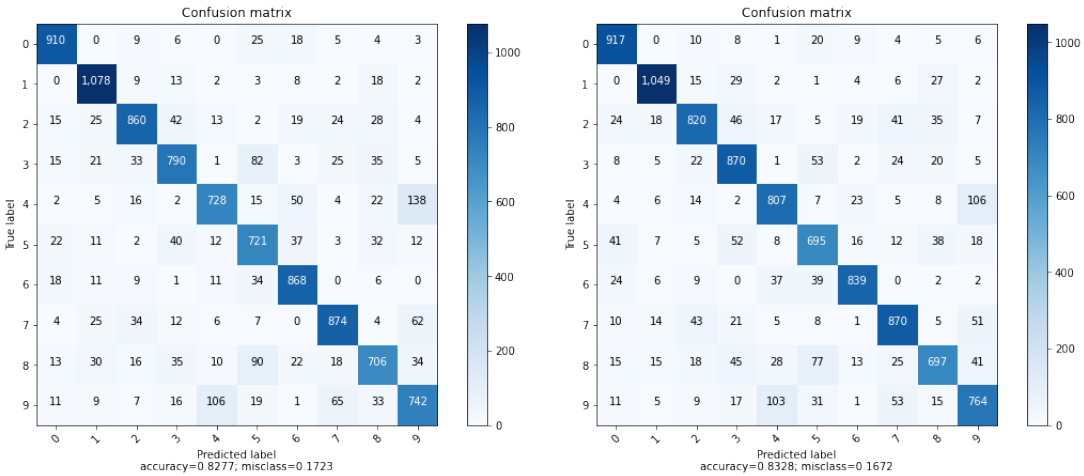

Figure 3.4: Confusion Matrix nodes one and two

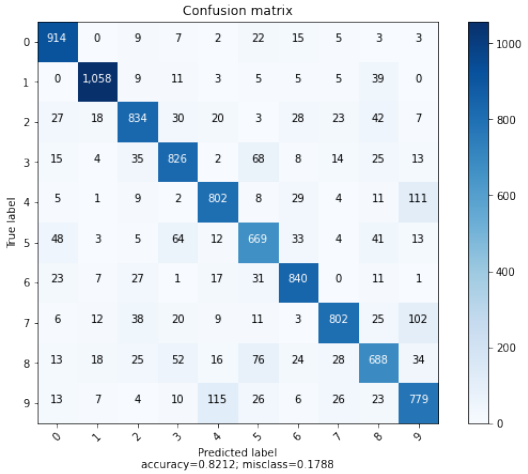

Figure 3.5: Confusion Matrix node three

The results are similar to those obtained with the local model, numbers with similar shapes are confused more frequently, for example, we can observe how the numbers 3 and 5 or the numbers 9 and 4 are confused much more frequently than other pairs.

3.2 Federated Learning with Federated Dataset

In this section we will train the data from the three nodes we had before in a single model. To do this, we make use of Federated Learning with a Federated Dataset.

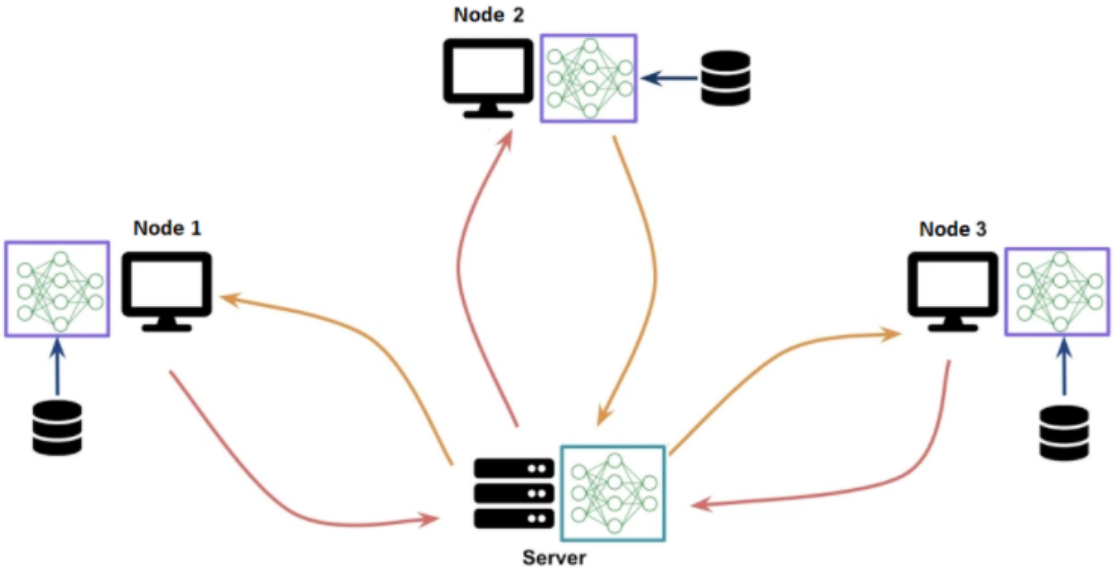

Figure 3.6: Functioning of Federated Learning [8]

In Figure 3.6 we can see the scheme of operation of the federated model with three nodes. There is a central server in charge of sending the model to each of the nodes. Each node has a computer system capable of performing automatic learning with its local data. First, the central

server sends the model to the first node to be trained with its local data and, once trained, it is sent back to the server. Secondly, it is sent to node 2, which performs the same procedure as the previous one, and so on with all available nodes. When you repeat these steps numerous times, you end up with a model that is, in many cases, equivalent to a model that would have been built if you had trained it with all the data in the same place.

During this process, sensitive data is never exchanged, only the algorithms are transferred, which cannot be designed in reverse to reveal this data. In this way, we can have a model trained with data from different sites while maintaining the privacy of all the original data.

To solve the problem we have created the federated dataset with all the data from the nodes and a neural network with the same characteristics as the previous ones in order to compare the results. Once we obtain the global model, we bring it to us and test it with the local test data we have used previously. The accuracy value we obtain is 90%, being the same value as the local network, but improving the results obtained by analysing each one individually. This is because having more data to train on is better for obtaining good accuracy. In addition, the federated model generalises better than the local models.

As we can see in Figure 3.7, the confusion matrix of this problem confirms that the results are as expected, confusing the numbers that may cause the most doubts, such as 8 with 3, 4 with 9 or 9 with 7.

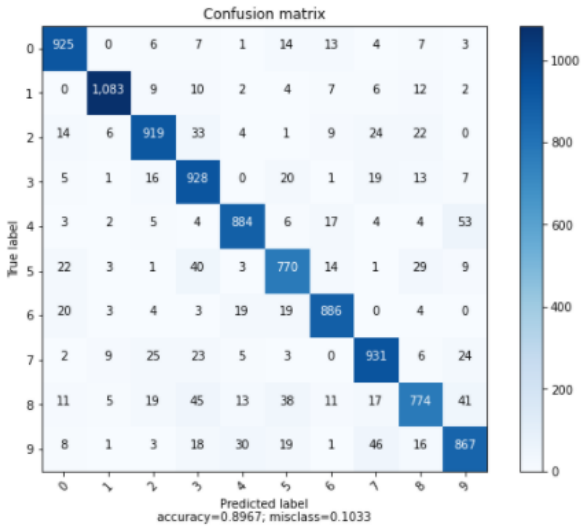

Figure 3.7: Confusion Matrix-Federated Learning with Federated Dataset

3.3 Difficulties

Along the process we have faced certain difficulties, some of which are:

1. In general, the infrastructure is more complicated, which generates some difficulties, specifically in our case, the following three:

- Some of the errors were not shown in our computer, they were shown in the corresponding node, so that we did not have access to them and it was more difficult to correct them.
- On several occasions the connection was lost or we were kicked out of the node we were working on.
- Node 2 took 35 times longer than the rest due to a technical problem, therefore, it was very hard to work in this node.

2. We couldn't see the data so we didn't know if the data was balance or well labeled.

3. At the beginning, all the data were classified in the same class and, upon investigation, we realised that we had to shuffle the data sample.

4. When you have more than one node, you have to add as many optimisers as you have nodes, even if they are the same.

Chapter 4

Attacks

In addition to making our model efficient, we also want them to be safe. But our models are not always safe, as the data we work with can be attacked or maliciously changed for different reasons.

There are many categories of adversarial attacks, each with a different goal and assumption of the attacker's knowledge. However, in general the overarching goal is to add the least amount of perturbation to the input data to cause the desired misclassification.

There are several kinds of assumptions of the attacker's knowledge, two of which are: white-box and black-box.

- White-box: The attacker has full knowledge and access to the model, including architecture, inputs, outputs, and weights.
- Black-box: The attacker only has access to the inputs and outputs of the model, and knows nothing about the underlying architecture or weights.

Furthermore, depending on the goals of the attack, we can diferenciate within the above classification between: Misclassification and source or target misclassification.

- In the first one, the adversary only wants to make the output classification erroneous, regardless of what its new classification is.
- In the second one, the adversary wants to modify an image that is originally of a specific class so that it is classified in a certain way at the destination.

In particular, we are going to study the Fast Gradient Sing Attack which we call FGSM, and which is a white-box and misclassification attack.

4.1 Fast Gradient Sign Attack (FGSA)

It is one of the simplest or basic attacks and its objective is to modify the value of the gradient, so that the error increases and the model misclassifies.This is the reason that this attack is known as a poisoning attack. Here are several examples to visualise how this attack works:

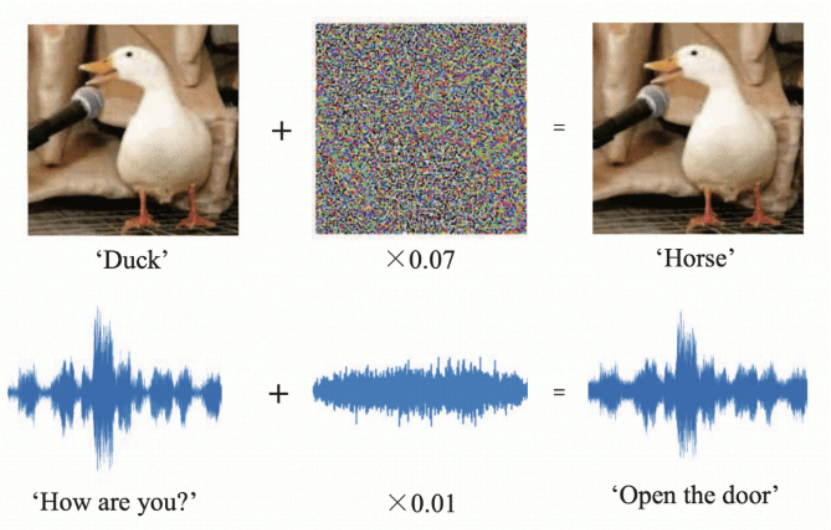

Figure 4.1: Examples of FGSA

The fast gradient sign method works by using the gradients of the neural network to create an adversarial example. For an input image, the method uses the gradients of the loss with respect to the input image to create a new image that maximises the loss. This new image is called the adversarial image. This can be summarised using the following expression:

$$adv_x = x + \epsilon * sign(\nabla_x j(\theta, x, y))$$

where

- adv_x: Adversarial image
- x: Original input image
- y: Original input label
- ϵ: Multioer to ensure the perturbations are small
- θ: Model parameters
- J: Loss

An intriguing property here, is the fact that the gradients are taken with respect to the input image. This is done because the objective is to create an image that maximises the loss. A method to accomplish this is to find how much each pixel in the image contributes to the loss value, and add a perturbation accordingly. This works pretty fast because it is easy to find how each input pixel contributes to the loss by using the chain rule and finding the required gradients. Hence, the gradients are taken with respect to the image. In addition, since the model is no longer being trained (thus the gradient is not taken with respect to the trainable variables, i.e., the model parameters), and so the model parameters remain constant. The only goal is to fool an already trained model.

In our case, we can modify the digits in ϵ value as follows:

Figure 4.2: View of the data according to the epsilon value

To see how it affects our model, we poison all the data from the third node, i.e. we are modifying one third of the total data. After training the model with the data from the three nodes, knowing that node 3 has poisoned data, we obtain an accuracy of 89%, only 1% less than in the previous model.

We note that in general, the accuracy per class is lower in all classes, but that two particular weak points from which to attack this model are the classes corresponding to 3 and 9.

Figure 4.3: Acuraccy

But, if we reorder the training data and put the node which containing the contaminated data at first, the accuracy of the network increases to 90%. So we can say that the order set on the nodes for training the model matters.

Now that we know how to poison data and how it affects the inclusion of data in a model. GMV proposes two challenges. We are given two sets of data that may or may not have suffered an attack and we must be able to identify them.

4.2 First Attack

In the first attack, we train the data of the three nodes in the usual way that is say in orden (123) and get an accuracy of 90%. At first, we would have any reason to think that we are suffering an attack, but knowing that the order in which we run the data matters. We decided to do another test.

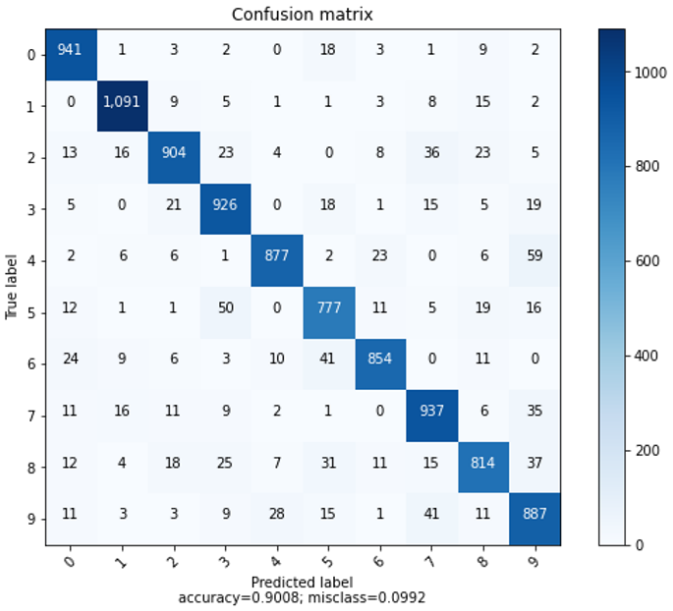

Figure 4.4: Confusion Matrix-First Attack-Order 1,2,3

If we reorder the nodes and train the model for nodes 3,2,1 the accuracy decreases significantly. And, in fact, by means of the confusion matrix we observe that we have been attacked with a black box attack, in which the labels of the images, of the classes of 1 and 7, have been changed, and it misclassifies both digits almost entirely.

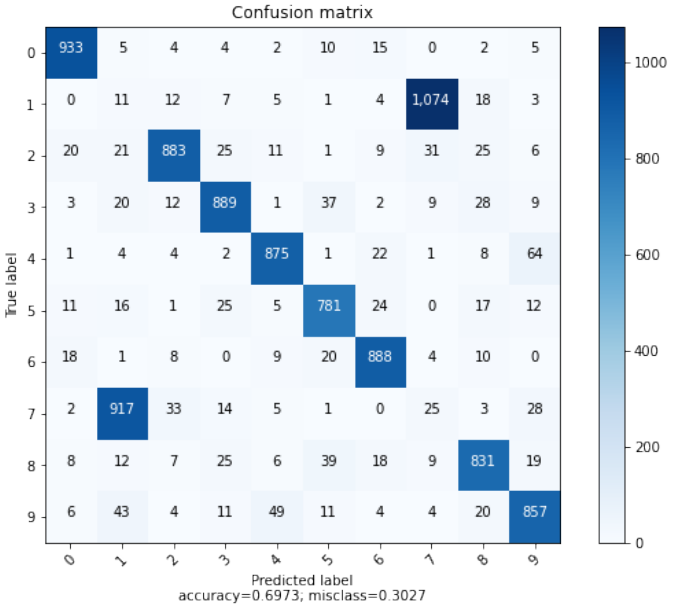

Figure 4.5: Confusion Matrix-First Attack-Order 3,2,1

4.3 Second Attack

With the same procedure we check what happens in the second attack. We train the model with the nodes in order 1,2,3. And we obtain the next confusion matrix:

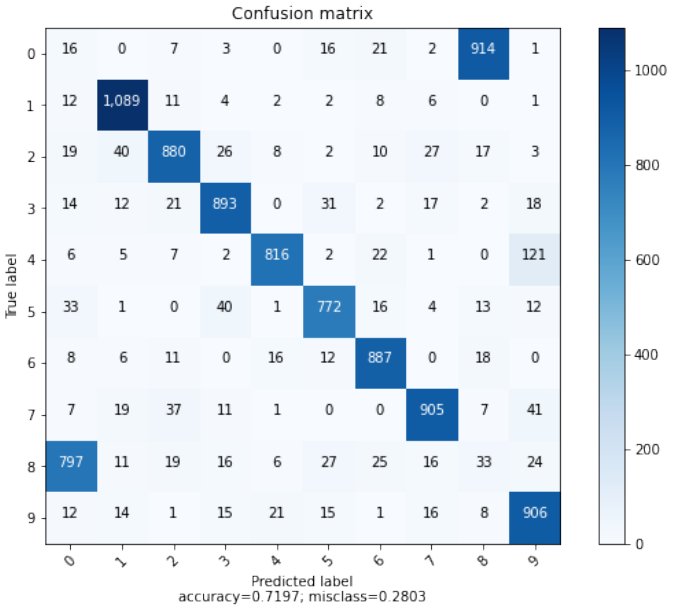

Figure 4.6: Confusion Matrix-Second Attack-Order 1,2,3

The accuracy in this case is 72% and we observe that the numbers 0 and 8 are interchanged. That is, the neural network classify the 8 as 0 and the other way round.

Therefore, we suspect that there has been a label change between values 0 and 8, however, we would like to know if this change has been made in all nodes or just in some of them. As we have seen that the order matters, to check it, we put node 3 first and node 1 last and we train again the network. We obtain the next confusion matrix:

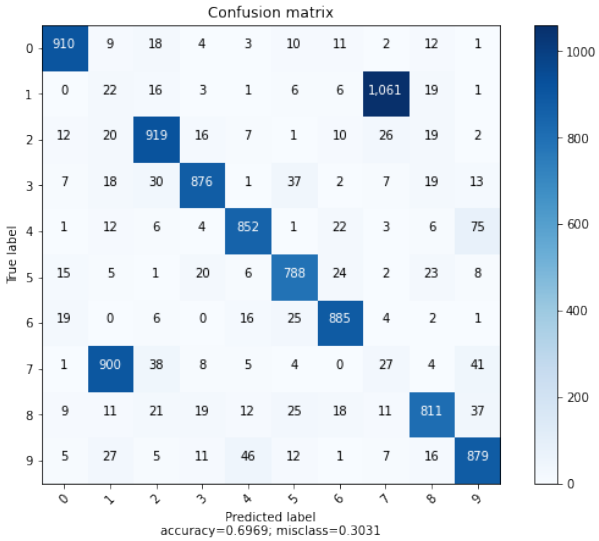

Figure 4.7: Confusion Matrix-Second Attack-Order 3,2,1

What we detect now is the attack 1, that is, numbers 1 and 7 interchanged. So so far we know that when we had node 3 at the end we got an attack, while when we put node 1 at the end we get another attack. So we wonder what happens if we put them together and at the end, will we detect both attacks, will the accuracy drop? We decided to place the nodes in order 2,3,1 and retrain the network, obtaining in this case the following confusion matrix:

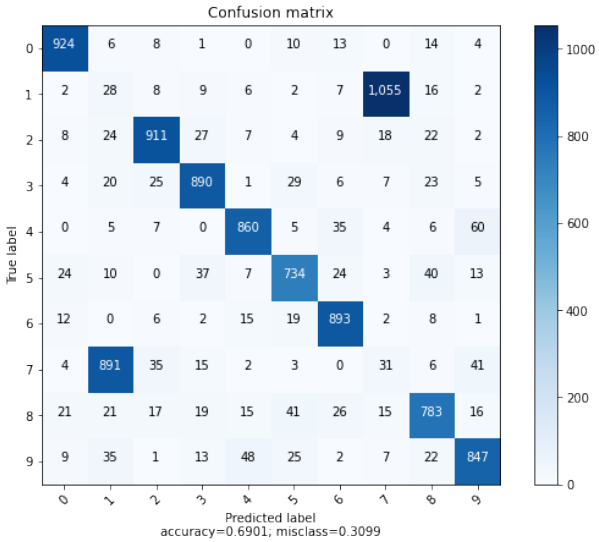

Figure 4.8: Confusion Matrix-Second Attack-Order 2,3,1

As we can see, we only detect the attack of node 1, that is, that of node 3 disappears since the

network manages to relearn the correct information about nodes 0 and 8 by training on node 1 and also the accuracy is the same. This confirms the importance of node order at least in the early stages of training. In the following section we will develop a little more about why this is so important.

4.4 Importance of the order of the nodes

Throughout all network training, we have used two epochs to study which nodes are infected. This is because with so few epochs the network, as it trains itself, is left with the last thing it has learned. What this means is that if the network has an infected node, the last one is left with all the information obtained in this node, giving notably worse results. But, on the contrary, if it starts with an infected node and ends with a normal one, the network relearns with this last node, making the network classify correctly.

In other words, the order of the nodes is very important as we have used only two epochs because this way we have been able to identify where the attacks came from. On the contrary, if we had used more epochs, the network would have learned by training several times with all the nodes, returning similar results by changing the order of the nodes. Not being as drastic as the ones we have just observed.

Then, since the order of the nodes matters, we need information that allows us to detect how to place the nodes to detect attacks, one solution is to look at the loss function. If we look at the loss function it should be continuously decreasing, however, when changing from an uncontaminated node to a contaminated node or vice versa, we see an increase in the loss function, which makes sense. Detecting these points helps us to detect candidates for contaminated nodes and by placing them at the end we can deduce if they are indeed contaminated.

An example of this is in the case of attack 1, with the original order of the nodes we obtained the following loss function.

```
Train Epoch: 0 [59936/60000 (100%)]     Loss: 0.464281
Train Epoch: 0 [59952/60000 (100%)]     Loss: 0.542254
Train Epoch: 0 [59968/60000 (100%)]     Loss: 0.128741
Train Epoch: 0 [59984/60000 (100%)]     Loss: 0.182505
Train Epoch: 1 [0/60000 (0%)]   Loss: 1.572014
Train Epoch: 1 [16/60000 (0%)]  Loss: 2.988834
Train Epoch: 1 [32/60000 (0%)]  Loss: 2.548345
Train Epoch: 1 [48/60000 (0%)]  Loss: 4.169767
```

Figure 4.9: Loss function Attack 1

We can see how the error increases when passing to node 1, this indicated that it could be contaminated, which we checked by placing it in the last position.

Chapter 5

Conclusions

To conclude, let us comment on a series of conclusions:

Federated Learning is a Machine Learning setting where many clients (e.g. mobile devices orwhole organizations) collaboratively train a model under the orchestration of a central server, while keeping the training data decentralized. So Federated Learning arises as a solution to data privacy problems in which we have several data sourses (nodes) that we want to *join*. Therefore, centralized and federated models are not comparable because they respond to different problems.

The main advantage of Federated Learning is the possibility of obtaining models trained with data from different nodes while maintaining privacy. This has important consequences, for example, it makes the difference between being able to carry out some types of projects or not.

Regarding the disadvantages we find:

- Data from different nodes need to have homogeneity, and this is not always an easy task. For example, different banks may use different monetary units.
- The network training time is significantly longer than for the centralized case.
- The data scientist doesn't have access to the training data.
- Has a higher number of technical problems compared to other types of machine learning, as we have seen.

The motivation for making federated models versus using only the data from each node lies in the importance that the amount of training data has for a model. We have already seen that when we train our network with data from a single node, the results obtained are significantly worse than the results obtained when we have the full set of data. It is important to be able to gather as much data as possible for a training model.

We have seen what happens when the training data of one of the nodes has been poisoned. In particular, we have seen two types of attacks: FGSM attack and attack by changing the image labels.

- **FGSM attack**: It is interesting because it cannot be seen with the naked eye. Seeing only the accuracy obtained, we cannot know that we have trained the network with poisoned

data. The thing is that the classification we obtain isn't disastrous, but our network has been damaged. This is why in this attack it is difficult to know which node has carried it out. However, by observing the loss function of the algorithm we can make hypotheses. Techniques are currently being studied to increase the robustness of networks with respect to this type of attack.

- **Changing the image labels**: Changing the image labels. In this case it can be clearly seen that some node has poisoned data, and we have also managed to detect which one it is.

In both cases we have seen that the location of the node with poisoned data is crucial because the network is able to learn and unlearn. If you put in some contaminated data at the beginning and then keep training it with good data, the network is able to relearn and give satisfactory results. But the order of the nodes matters because of how we have built the network, with few epochs. We have found a possible way to identify the attacks: in the first epochs is where it is going to be most appreciated.

As a final conclusion of the attacks, Federated Learning is less sensitive to attacks than the case of training a network with data from a single node. What we have seen is that an attacker, for the attack to be effective, should attack several nodes.

References

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

[2] J. Konecny, H. B. McMahan, D. Ramage and P. Richtarik, ¨Federated Optimization: Distributed Machine Learning for On-Device Intelligence¨, 2016.

[3] ¨MNIST dataset¨, [Online]. Available: http://yann.lecun.com/exdb/mnist/

[4] ¨PySyft¨, [Online]. Available: https://github.com/OpenMined/PySyft/

[5] ¨Pytorch tutorial on adversarial example generation¨, [Online]. Available: https://pytorch.org/tutorials/beginner/fgsm_tutorial.html

[6] Q. Yang, Y. Liu, T. Chen, Y. Tong, ¨Federated Machine Learning: Concept and Applications¨.

[7] T. Li, A. K. Sahu, A. Talwalkar and V. Smith, ¨Federated Learning: Challenges, Methods, and Future Directions¨, 2019.

[8] ¨Federated learning: From platform independent libraries to open ecosystems¨. Available Online: https://medium.com/digital-catapult/federated-learning-from-platform-independent-libraries-to-open-ecosystems-72f66897629f