TRAVAUX D'ÉTUDES ET DE RECHERCHE

DEEP LEARNING FOR AUTONOMOUS CARS
FEDERATED LEARNING

Hadrien BONATO-PAPE, Hossein FIROOZ,

Vincent LAUBRY, Erdal TOPRAK

Février 2022

Contents

1 Introduction

1.1 Presentation of the group

This project is supervised by :

- Diane LINGRAND
- Frederic PRECIOSO

Our project group is composed of four members :

- Hadrien BONATO-PAPE (SI5-WIA)
- Hossein FIROOZ (M2-EIT Digital)
- Vincent LAUBRY (M2-IoTCPS)
- Erdal TOPRAK (M2-SD)

Despite our differences in our specialities we all have some background in machine learning.

Hadrien studied at the Faculty of Sciences of Toulon in Electronics and Computer Science during two years. And since 3 years he study computer science at Polytech Nice Sophia, specialized in Web and IA. He have mainly worked for the French Army and for Cap Gemini, and "fostr", his own company.

Hossein is part of EIT Digital Master School which is a dual Master's degree program funded by European Union. His major is Computer Science and he is getting a minor in business innovation as well. Previously he was affiliated with Aalto University in Finland. Hossein maily has experiences on Reinforcement Learning and its applications on Autonomous Systems.

Vincent studied at the Faculty of Sciences of Valrose. His first two years of study were in mathematics and computer science, his third in computer science only, his fourth in a Master's degree in computer science and finally, his fifth year currently at Polytech in IoT-CPS.

Erdal studied at the Faculty of Sciences of Valrose. His first two years of study were in mathematics and computer science, his third in computer science and his fourth and fifth in a Master's degree in computer science specialized in data science.

1.2 Presentation of the subject

Our subject, named ”Deep Learning for Autonomous Cars / Federated learning, inter-object communication”, is one of the two sub-projects for autonomous cars.
The first sub-project has for objective to make multiple computer vision models coexist together in the most efficient way. Our sub-project has for objective to implement federated learning, a distributed way of doing machine learning, to compute constrained edge devices.

The project is placed in the context of technologies used by contemporary autonomous vehicles.
By their nature, these vehicles have a limited computing capacity and choices must be made in order to optimize the use of machine learning models. This is part of a realistic approach: Where the computational performance of on-board systems evolves rapidly, these performances quickly reach limits, because of energy consumption or cooling constraints for example.

The goal of our project is to decentralize this machine learning in order to reduce the weight of the embedded systems in autonomous vehicles. Moreover, it allows a certain "communication" between vehicles, which allows them to collaborate in the management of events that occur to them.

In order to represents the autonomous vehicles, we are using Sunfounder Picar-X, a small chassis equipped with a camera, a locomotion system, a steering system, and based around a Raspberry Pi 4. The Raspberry Pi 4 is equipped with a quad-core Cortex-A72 processor and 4GB of RAM in our case. It is easily understandable that these Raspberry Pi 4 constrain us in terms of computing power.

Always with the aim to place these small cars in a context close to reality, a small track has been created especially for this. It is located at the Maison de l’Intelligence Artificielle in Sophia-Antipolis, which kindly welcomed us for our project.

On this track, some traffic signs may be placed. The signs that have been created are:

- Speed limit 30 km/h
- Speed limit 50 km/h
- Stop sign
- Wrong way

Figure 1: Track at the "Maison de l’Intelligence Artificielle"

2 State of the art

2.1 Technical concepts

2.1.1 The Autonomous Vehicle

Figure 2: Level 4 of the Autonomous Vehicle

We can define an autonomous vehicle by being a vehicle that is aware of it's environment and can operate without human intervention. According to the NHTSA[1] there are 5 levels of car autonomy:

LEVELS OF AUTOMATION	WHO DOES WHAT, WHEN
Level 0	The human driver does all the driving.
Level 1	An advanced driver assistance system (ADAS) on the vehicle can sometimes assist the human driver with either steering or braking/accelerating, but not both simultaneously.
Level 2	An advanced driver assistance system (ADAS) on the vehicle can itself actually control both steering and braking/accelerating simultaneously under some circumstances. The human driver must continue to pay full attention ("monitor the driving environment") at all times and perform the rest of the driving task.
Level 3	An automated driving system (ADS) on the vehicle can itself perform all aspects of the driving task under some circumstances. In those circumstances, the human driver must be ready to take back control at any time when the ADS requests the human driver to do so. In all other circumstances, the human driver performs the driving task.
Level 4	An automated driving system (ADS) on the vehicle can itself perform all driving tasks and monitor the driving environment – essentially, do all the driving – in certain circumstances. The human need not pay attention in those circumstances.
Level 5	An automated driving system (ADS) on the vehicle can do all the driving in all circumstances. The human occupants are just passengers and need never be involved in driving.

Figure 3: NHTSA car autonomy levels

2.1.2 IoT

The Internet of Things (IoT) is the interconnection between the Internet and physical objects, places and environments of our world. The name refers to a growing number of objects connected to the Internet, allowing communication between our so-called physical assets and their digital existence.

According to Oracle[2], IoT describes the network of physical devices, the "objects", that uses sensors, software and other technologies to connect themselves to other devices and systems on the Internet and exchange data with them.

2.1.3 Federated Learning

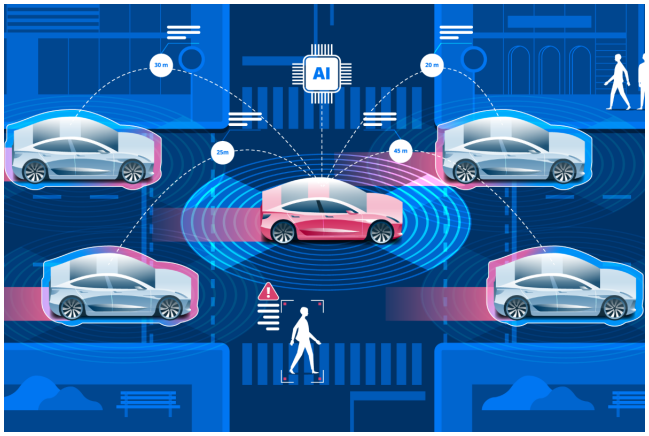

Figure 4: Communication between vehicles

According to the Google Blog[3], Federated Learning enables mobile phones to collaboratively learn a shared prediction model while keeping all the training data on device, decoupling the ability to do machine learning from the need to store the data in the cloud. This goes beyond the use of local models that make predictions on mobile devices by bringing model training to the device as well.

It works like this: your device downloads the current model, improves it by learning from data on your phone, and then summarizes the changes as a small focused update. Only this update to the model is sent to the cloud, using encrypted communication, where it is immediately averaged with other user updates to improve the shared model. All the training data remains on your device, and no individual updates are stored in the cloud.

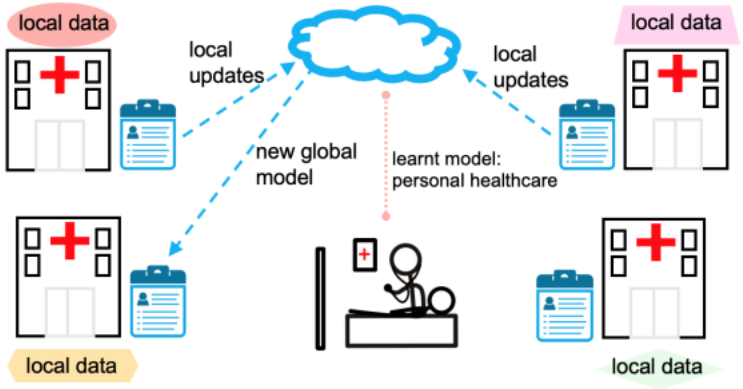

Figure 5: Visualisation of Federated Learning (Carnegie)

2.2 Car Presentation

2.2.1 Picar-x

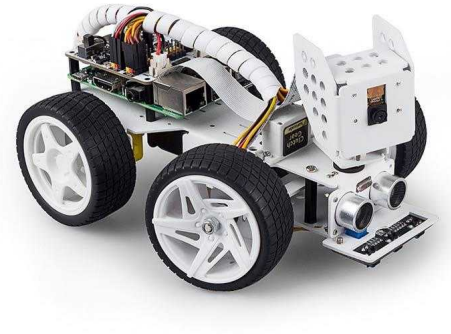

Figure 6: Picar-x

In order to get as close as possible to reality to simulate a car, we use the Picar-x for our project. This small car is provided by SunFounder. The PiCar-X is an AI self-driving robot car for Raspberry Pi, on which RPi works as the control center. There are camera module, ultrasonic module, line tracking module mounted on it.

2.2.2 Raspberry Pi 4

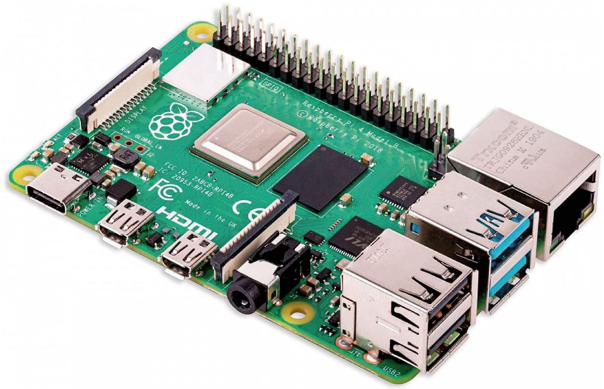

Figure 7: Raspberry Pi 4

The Raspberry Pi is a low cost, credit-card sized computer that plugs into a computer monitor or TV, and uses a standard keyboard and mouse. Every new generation of the Raspberry Pi offers more power and memory, which allow us to do biggest implementations. For our work, we I mainly used with wifi connected by ssh from our computer. The Picar-x is a client a for our federated learning.

2.3 Frameworks and Algorithms

2.3.1 Flower

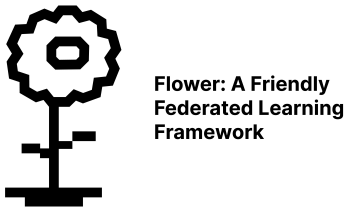

Figure 8: Flower

The Flower Federated Learning framework offers an approach to federated learning. With it, you can federate any workload, any ML framework, and any programming language. According to the framework own description[18], the design of Flower is based on a few guiding principles:

- Customizable: Federated learning systems vary wildly from one use case to another. Flower allows for a wide range of different configurations depending on the needs of each individual use case.
- Extendable: Flower originated from a research project at the Univerity of Oxford, so it was build with AI research in mind. Many components can be extended and overridden to build new state-of-the-art systems.
- Framework-agnostic: Different machine learning frameworks have different strengths. Flower can be used with any machine learning framework, for example, PyTorch, TensorFlow, Hugging Face Transformers, PyTorch Lightning, MXNet, scikit-learn, TFLite, or even raw NumPy for users who enjoy computing gradients by hand.
- Understandable: Flower is written with maintainability in mind. The community is encouraged to both read and contribute to the codebase.

2.3.2 PyTorch

Figure 9: PyTorch

PyTorch doesn't have an official implementation for the federated learning based on [19]. However by separating the data in a way that data points are IID, Independent and Identically Distributed, the averaging algorithms explained in next section can be also implemented via PyTorch. In part 3.2.4 we will see how to implement this framework

2.3.3 Avg Algorithm

In the federated learning, there are parallel models training and we have to combine the gradients. Flower and our PyTorch implementation both uses Google's Federated Averaging algorithm [19]. In this approach, parallel models are training using IID datasets and after each epoch, a fraction of clients are selected, and based on those, we compute the gradient of the loss over all the data held by these clients. Therefore,this fraction is a proxy for controlling the global batch size. With C = 1 the algorithm is equivalent to full-batch (non-stochastic) gradient descent.

Extensions to this algorithm allow the dataset to be non-IID as well, however in our PyTorch implementation as well as the default configuration of Flower, we assume the data is IID.

2.3.4 TensorFlow FL

Figure 10: TensorFlow

TensorFlow Federated Learning is a framework to easily doing federated learning. The basic use of Tensor Flow Federated Learning is to create clients with the same original model. After training them, there is the aggregation step, which consists of mixing the models. This aggregation is performed using the "Avg Algorithm" algorithm. This framework is of course designed to work with keras models. And finally there is the redistribution of models to different clients. In part 3.2.2 we will see how to implement this framework.

3 Realised work

3.1 Experiments

3.1.1 Local implementation of Flower

After our research, we have found that Flower was a good Framework to do Federated Learning.
So we decided to start to understand this framework and implement something with it by starting to do the tutorial.

The first implementation was to train a Convolutional Neural Network on CIFAR10 using Flower and PyTorch. It consists of one server and two clients all having the same model.
Clients are responsible for generating individual weight-updates for the model based on their local datasets. These updates are then sent to the server which will aggregate them to produce a better model. Finally, the server sends this improved version of the model back to each client. A complete cycle of weight updates is called a round.

Our training procedure and network architecture are based on PyTorch's Deep Learning with PyTorch. We use PyTorch to load CIFAR10, a popular colored image classification dataset for machine learning.
The Flower server interacts with clients through an interface called Client. When the server selects a particular client for training, it sends training instructions over the network.
The client receives those instructions and calls one of the Client methods to run your code (i.e., to train the neural network we defined earlier).

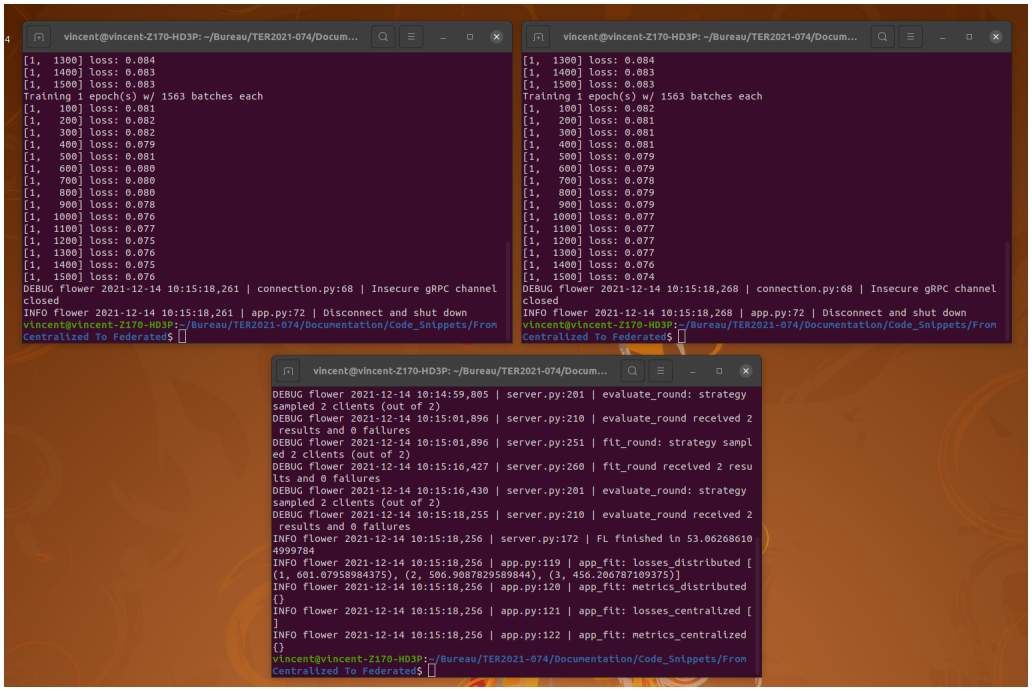

Figure 11: TensorFlow

3.1.2 Local implementation of TensorFlow FL

To test this framework, we used the tutorial notebook written by the developers of TensorFlow. Available at: https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification?hl=en

The experiment is very simple, it is a question here of randomly sending pieces of the MNIST dataset (representing the handwritten numbers from 0 to 9).

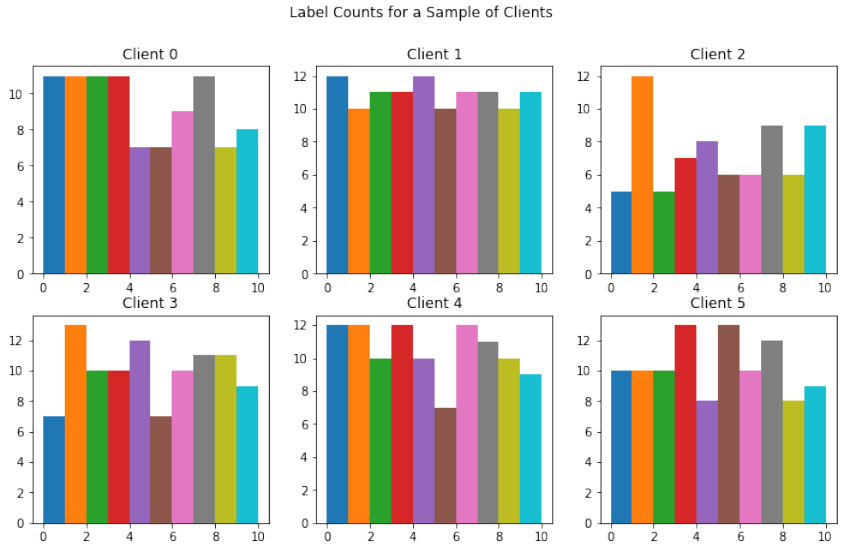

Figure 12: Data distribution between the clients.

The objective is after training to collect the models and send a better model to each client. At the first aggregation the model does not have good results, only 0.118 of accuracy. But after 30 aggregations and training we get an accuracy of 0.8255144. As we can see below :

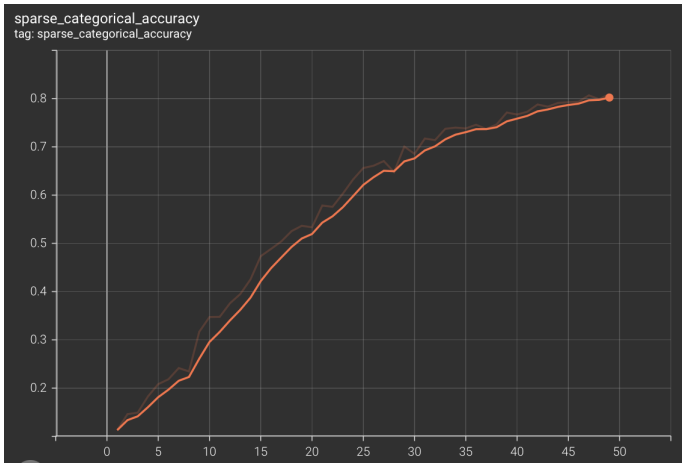

Figure 13: Accuracy per training/aggregations.

TensorFlow FL is such a good and easy way to do federated learning, but a bit complicated compared to Flower framework.

3.1.3 Local implementation of PyTorch FL

In our experiments, we trained 3 separated models using $P5000$ GPU on MNIST dataset where the data for each client was IID. For this approach, we dived the dataset in three parts.

Although there are standard benchmarks for benchmarking federated learning algorithms [20], here we used accuracy of the cumulative model and the training time. Our base-model was training MNINST dataset via the following 2 layer CNN model:
We used the following parameters in Table 1 for our basemodel:

Table 1: Baseline Model Hyperparameters

Optimizer	**Learning Rate**	**Batch Size**
SGD	0.01	16

```
net = CNNFashion_Mnist()
from torchsummary import summary
summary(net.to('cuda:0'), (1, 28, 28))
```

```
----------------------------------------------------------------
        Layer (type)               Output Shape         Param #
================================================================
            Conv2d-1           [-1, 16, 28, 28]             416
       BatchNorm2d-2           [-1, 16, 28, 28]              32
              ReLU-3           [-1, 16, 28, 28]               0
         MaxPool2d-4           [-1, 16, 14, 14]               0
            Conv2d-5           [-1, 32, 14, 14]          12,832
       BatchNorm2d-6           [-1, 32, 14, 14]              64
              ReLU-7           [-1, 32, 14, 14]               0
         MaxPool2d-8             [-1, 32, 7, 7]               0
            Linear-9                   [-1, 10]          15,690
================================================================
Total params: 29,034
Trainable params: 29,034
Non-trainable params: 0
----------------------------------------------------------------
Input size (MB): 0.00
Forward/backward pass size (MB): 0.47
Params size (MB): 0.11
Estimated Total Size (MB): 0.58
----------------------------------------------------------------
```

Figure 14: MNIST CNN model

Table 2: Federated Learning vs Baseline model Accuracy

Model	**Test Dataset Accuracy**	**Training Time**
Baseline Model	**98.13**	∼12 Min
Federated Model with 3 clients	97.65	**∼5 Min**

Table 2 shows the results for both trainings

Based on the results, we can see our baseline model has a better accuracy. Accuracy is higher as all the data is located in one machine and the model has access to all the data. Lower accuracy is expected in federated models as each model doesn't have access to all the data and only observe the dataset partially.
Lower training time is also expected in our implementation for federated learning as we train the cumulative model for only 2 epochs and we are training client networks in parallel.
Therefore we can argue using federated learning, we can achieve a high accuracy with lower training time.

3.1.4 Implementation of Flower on Picar-x

To experiment Federated Learning on cars we should have a use case.

Our first use case was to train the model that allows the car to move between lines via the camera. Our experiment would have been simple, two vehicles, one on each side of the track, they both learn part of the track, then we aggregate the models. And we swap cars. Following this, they should be able to evolve perfectly on an unexplored area, with new types of obstacles. Simply because of federated learning that shares model knowledge. However, the available code was absolutely not suitable for federated learning. So unless we completely rewrite an algorithm to evolve on the track, it is impossible to implement this use case.

A second use case then came to us, less realistic of course, but it allows us to demonstrate the usefulness and functioning of federated learning. Our use case: The ability to recognize road panels on the track. A vehicle that would evolve in an area with mainly 50 panels, and very few 30 panels. The other vehicle would evolve in an area mainly made up of 30 panels, with few 50 panels.

We first had to develop a simple model to classify traffic signs (30 speed limit or 50 speed limit). Then an image is taken by the vehicle. We can spot the round objects in the image, and therefore the traffic signs. We cut the object in a sub-image that we save.

Once the data has been acquired, it must be augmented artificially. In order to obtain a sufficient number of examples to provide to the model for training, validation and testing of the model. The data is increased, by randomly zoom the image and randomly change its inclination. For a given image, 10 others are created. That is :

- 22 frames of 30 panels increased to 242 frames.
- 19 frames of 50 panels increased to 209 frames.

In order not to have to perform the augmentation each time, we save the images.

As a benchmark and to validate the operation of our CNN, we will train it with all the data (70% for learning, and 30% for testing).

```
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 conv2d_4 (Conv2D)           (None, 222, 222, 32)      896

 max_pooling2d_4 (MaxPooling  (None, 221, 221, 32)     0
 2D)

 flatten_4 (Flatten)         (None, 1562912)           0

 dense_4 (Dense)             (None, 2)                 3125826

=================================================================
Total params: 3,126,722
Trainable params: 3,126,722
Non-trainable params: 0
_________________________________________________________________
```

Figure 15: Model to classify road signs

Here are the results after 20 epochs:

- F1 score: 0.9322033898305084
- Accuracy: 0.9407407407407408

3.2 Results

Let's move on to adapting the CNN on Raspberry Pis. Our experience requires 2 clients and 1 flower server. In local we have trained and saved the models for the two clients (1 and 2). The models are the same but trained with different data. The pre-trained models for each clients are loaded on both clients. With this step we avoid the first long training on the Raspberry Pis.

The models are pretrained with : - Client model 1 has been trained 176 images of 30 panels, and 11 images of 50 panels. - Client model 2 has been trained 11 images of 30 panels, and 143 images of 50 panels. - The test dataset common to both clients has 55 images of each class.

After having loaded the models for each client we check the results. And accordingly to the local experiments : before the aggregation of the two models the results are very bad. Details :

- Client 1 (trained with a majority of 30 panels): F1 score = 0.0
- Client 2 (trained with a majority of 50 panels): F1 score = 0.67

After 3 new aggregations the results are better, even if not yet perfect.

Client 1 (trained with a majority of 30 panels):

- F1 score : 0.67
- Accuracy : 0.62

Client 2 (trained with a majority of 50 panels):

- F1 score : 0.67
- Accuracy : 0.62

We have the exactly same results, it's normal, the models are the same now.

In conclusion, like in the local experiments federated learning works well, even if it remains perfectible. The first reason for the poor results is the lack of data. We need more data to improve our models. But we have proof that federated learning helps to share knowledge to create a single model bringing together all clients data.

A little problem of the Raspberry Pis is the lack of computing power, so it's such a long process to test the models after the aggregations on each Raspberry Pis.

4 Project Management

According to PMI[21], project management is the use of specific knowledge, skills, tools and techniques to deliver something of value to people. For example the development of software for an improved autonomous car environment.

Once the subject was validated by our teachers, we began to implement a proper working and development environment.

4.1 Human resources management

As a group of 4 students we separated the tasks according to our specialities and also according the various issues we identified. We all focused on the first two weeks understanding the concept of federated learning. Then we focused on implementing a local proof of concept and integrating and optimising it for edge devices.

4.2 Communication tools

Since we are still in a pandemic period, we relied a lot on remote communication and work. The communication tools that we relied on are asynchronous and searchable in order to work with our different schedules.

For our communications we relied on Discord, a centralised group service. In our private group we split the communications into different channels, for example, channels dedicated to resources and live communications. While remote communications were the most frequent we met several times at "La maison de l'Intelligence Artificielle" to take our important decisions and test our work.

4.3 Development tools and cycle

To collaborate on our code with set up a GitHub repository and divided the development cycle in 3 parts:

- Identifying a feature and associating it with a task / subtask
- Implementing a local version
- Deploying the version into the Picar-X

5 Conclusion

5.1 Summary

This project was for our group, a milestone in the comprehension of federated learning and more broadly privacy challenges in data science.

To accomplish our goals we first read, analysed and tried to understand federated learning. This important step was the key to implement a viable architecture. Federated learning is one of the many ways we can distribute the computing resources while maintaining data ownership.

So far, we implemented an extensible architecture based on the latest frameworks that allows us to have several clients that perform federated learning. To demonstrate our results we implemented our own pre-trained model into the Raspberry Pis and observed the outputs as the knowledge of each cars improved.

At the end, we were able to converge the knowledge of two cars to each other through federated learning. With an accuracy of 0.62, this is the first of many steps to come in order to have a real car with this type of architecture.

5.2 Reflections on our work

While we didn't encounter problems in the scope of this projects, the challenges in terms or privacy and real-time computation are plentiful and not well understood in order to apply our deployment in a production environment. For example, a common inter-car communication protocol should be established in order to maximize the efficiency of federated learning and avoid unnecessary computation.

Moreover, federated learning also has some competition as distributed learning becomes more and more a viable privacy and compute efficient alternative.

5.3 Acknowledgement

Finally, we would like to thank our teachers, Diane LINGRAND and Frédéric PRECIOSO, for their dedication, guidance and the time accorded to us during this project. We would also like to thank "La Maison de l'Intelligence Artificielle" for providing us with a good working environment and also all the previous and future students that will work on this project.

6 Bibliography

[1] Definition of Automated Vehiciles *from NHTSA*. https://www.nhtsa.gov/technology-innovation/automated-vehicles-safety

[2] Definition of IoT *from Oracle*. https://www.oracle.com/internet-of-things/what-is-iot/

[3] Definition of federated learning *from Google*. https://ai.googleblog.com/2017/04/federated-learning-collaborative.html

[4] Federated learning page *from Wikipedia*. https://en.wikipedia.org/wiki/Federated_learning

[5] Federated learning documentation page *from TensorFlow*. https://www.tensorflow.org/federated/federated_learning

[6] Federated learning example for image classification *from TensorFlow*. https://www.tensorflow.org/federated/tutorials/federated_learning_for_image_classification

[7] Li, T., Hu, S., Beirami, A., and Smith, V. (2020). *Federated multi-task learning for competing constraints.* arXiv preprint arXiv:2012.04221. https://research.fb.com/wp-content/uploads/2021/05/Federated-Multi-Task-Learning-for-Competing-Constraints.pdf

[8] Smith, V., Chiang, C. K., Sanjabi, M., and Talwalkar, A. (2017, December). *Federated multi-task learning.* In Proceedings of the 31st International Conference on Neural Information Processing Systems (pp. 4427-4437). http://papers.neurips.cc/paper/7029-federated-multi-task-learning.pdf

[9] Zhu, H., Zhang, H. and Jin, Y. *From federated learning to federated neural architecture search: a survey.* Complex Intell. Syst. 7, 639–657 (2021). https://doi.org/10.1007/s40747-020-00247-z https://link.springer.com/content/pdf/10.1007/s40747-020-00247-z.pdf

[10] Tengchan Zeng, Student Member, IEEE, Omid Semiari, Member, IEEE, Mingzhe Chen, Member, IEEE, Walid Saad, Fellow, IEEE, and Mehdi Bennis, Fellow, IEEE *Federated Learning on the Road: Autonomous Controller Design for Connected and Autonomous Vehicles* arXiv:2102.03401v1 [eess.SY] 5 Feb 2021. https://arxiv.org/pdf/2102.03401.pdf

[11] Anh Nguyen, Tuong Do, Minh Tran, Binh X. Nguyen, Chien Duong, Tu Phan, Erman Tjiputra, Quang D. Tran *Deep Federated Learning for Autonomous Driving* arXiv:2110.05754v1 [cs.LG] 12 Oct 2021. https://arxiv.org/pdf/2110.05754.pdf

[12] Krunal Kshirsagar : Research Engineer And Writer @OpenMined *MAKING AUTONOMOUS VEHICLES ROBUST WITH ACTIVE LEARNING, FEDERATED LEARNING AND V2X COMMUNICATION*. Article posted on September 3rd, 2021. https://blog.openmined.org/making-autonomous-vehicles-robust-active-learning-federated-learning-v2x/

[13] Virginia Smith, Chao-Kai Chiang, Maziar Sanjabi, Ameet Talwalkar *Federated Multi-Task Learning*. 31st Conference on Neural Information Processing Systems (NIPS 2017), Long Beach, CA, USA. https://proceedings.neurips.cc/paper/2017/file/6211080fa89981f66b1a0c9d55c61d0f-Paper.pdf

[14] Xinchi Qiu, Titouan Parcollet, Javier Fernandez-Marques, Pedro, P. B. Gusmao, Daniel J. Beutel, Taner Topal, Akhil Mathur, Nicholas D. Lane *A First Look into the Carbon Footprint of Federated Learning*. https://arxiv.org/abs/2102.07627v3

[15] Gábor Szegedi, Péter Kiss, and Tomáš Horváth *Evolutionary federated learning on EEG-data*. http://ceur-ws.org/Vol-2473/paper14.pdf

[16] Lynda Ferraguig, Yasmine Djebrouni, Sara Bouchenak, Vania Marangozova *Survey of Bias Mitigation in Federated Learning*. Conférence francophone d'informatique en Parallélisme, Architecture et Système, Jul 2021, Lyon (virtuel), France. hal-03343288 https://hal.archives-ouvertes.fr/hal-03343288

[17] Shadi A. Noghabi, Landon Cox, Sharad Agarwal, Ganesh Ananthanarayanan *The Emerging Landscape of Edge-Computing*. GetMobile: Mobile Computing and Communications Volume 23 Issue 4 December 2019 pp 11–20. https://doi.org/10.1145/3400713.3400717 - https://dl.acm.org/doi/10.1145/3400713.3400717 - https://www.microsoft.com/en-us/research/uploads/prod/2020/02/GetMobile__Edge_BW.pdf - https://www.researchgate.net/publication/345139045_The_Emerging_Landscape_of_Edge_Computing

[18] Flower Framework for Federated Learning https://flower.dev/

[19] H. Brendan McMahan and Eider Moore and Daniel Ramage and Seth Hampson and Blaise Agüera y Arcas, Communication-Efficient Learning of Deep Networks from Decentralized Data, 2017, arXiv

[20] Sebastian Caldas and Sai Meher Karthik Duddu and Peter Wu and Tian Li and Jakub Konečný and H. Brendan McMahan and Virginia Smith and Ameet Talwalkar, LEAF: A Benchmark for Federated Settings, 2019, arXiv

[21] Definition of project management *from PMI*. https://www.pmi.org/about/learn-about-pmi/what-is-project-management