POLITECNICO DI TORINO

Master's Degree in ICT for Smart Societies

Master's Degree Thesis

Data-Driven Road Hazard Detection for Automated Driving

Advisor

Prof. Carla Fabiana CHIASSERINI

Co-advisor

Prof. Jérôme HÄRRI

Candidate

Giuseppe DI GIACOMO

Academic year 2020-2021

Summary

Recently, Machine Learning and, more specifically, Deep Learning have become the state-of-the-art techniques in different domains, such as computer vision, natural language processing and speech recognition. One of the main drawbacks is the necessity of a huge amount of data to be trained correctly. So far, the traditional way consists in gathering samples at only one central infrastructure, which trains the model. However, if data are recorded by different devices, this process presents two main disadvantages: first, transmitting all of them requires a great consumption of network resources and, second, it can expose sensitive information. In such a context, a new procedure has emerged: Federated Learning.

Basically, Federated Learning leverages the computational power of the agents collecting data, which, instead of uploading them to the central server in charge of the model training, keep samples locally and use them for the learning process. Iteratively, the server only aggregates all the received models.

ML and DL are being increasingly studied also in the transportation field, as they can be used for ADAS and autonomous driving. Other implementations are related to the intelligent transportation system, as these techniques may be applied, for instance, for travel time estimation, to decrease fuel consumption and for traffic optimization. Thanks to the many contexts in which it may be used, FL has gained interest also in vehicular networks.

In this work a novel method, named Hybrid Federated Learning, is introduced: with respect to the standard FL, users are gathered in groups. For each cluster, the server receives a model that has been sequentially trained by devices of the ensemble.

Compared to the vanilla FL algorithm, the presented approach shows better performances in terms of global iterations and number of transmissions. Also, the communication load on the central server is alleviated.

Table of Contents

List of Tables

List of Figures

Acronyms

ML Machine Learning

DL Deep Learning

FL Federated Learning

SFL Standard Federated Learning (referred to *FedAvg*

CNN Convolutional Neural Network

RNN Recurrent Neural Network

LSTM Long short-term memory

IID Independent and identically distributed

EMD Earth mover's distance

QoI Quality of information

WAN Wide Area Network

LAN Local Area Network

BS Base station

SGD Stochastic Gradient Descent

TV Target vehicle

EV Ego vehicle

MSE Mean squared error

MAE Mean absolute error

Chapter 1

Introduction

1.1 Federated Learning

With the technological advance in the last decades, phones, tablets, vehicles, wearable and IoT devices have become more powerful and provided with an increasing number of sensors: the amount of data collected by these machines is huge. For instance, an NVIDIA Deep Learning Data Scientist estimated in 2017 [1] that an autonomous vehicle with only 5 cameras can generate from 1 to 3 TB per hour of raw data; other researches estimate a volume of collected data between 4 and 20 TB per day, depending on the provided sensors [2] [3]. This is positive, as these data can be used for traditional Machine Learning and Deep Learning approaches, which need a lot of samples to be trained efficiently. However, transmitting such a quantity of information to the server in charge of the model training could lead to network bottlenecks, thus causing delays, and can be costly from a communication point of view.

Aside from this, also data security and privacy are now great concerns, reason for which laws related to the topic are getting more strict: the General Data Protection Regulation (GDPR) applied in the European Union in 2018 and the Consumer Privacy Bill of Rights, in the US, are a tangible proof about it [4].

Thus, it is natural to think of an approach that brings the intelligence closer to nodes, where input information is collected and produced, instead of transmitting data to a central processing infrastructure. This is done in Federated Learning (FL), a distributed machine learning paradigm, which consists in training a model in a decentralized manner.

Generally, to train a model, data are gathered by the central server and then used for the learning stage; after this process, the final model is deployed to the users to be utilized for inference. Furthermore, it is also possible to perform the training in a distributed way, exploiting different computers within the infrastructure: the

processing is shared across multiple units, but the decisions are still centralized and a complete system knowledge is exploited [5].

On the other hand, FL approach is still distributed, but also decentralized: the difference with respect to the previous method is that each node collects its own data, and instead of sending them to the central server, it keeps them locally and uses them to learn the model parameters [6]. However, a coordinator is still employed, but its function is limited, as it is only in charge of aggregating the received weights and transmitting the resulting model to devices participating to the successive iteration [6]. The number of aggregations depends on the quantity of global rounds performed during the training; furthermore, it is important to notice that at each round, users can process for one or more iterations, which are referred to as local epochs.

As said, FL prevents data from being transmitted over the network and exploits hardware of the devices for the processing. Due to these reasons, the main advantages of FL are:

- privacy and ownership, since data are not shared, but processed locally [7];
- computational power: as the number of involved machines can be high, the overall computational power of the federated network can be huge; added to this, devices are provided with increasing processing capacity[8].

By contrast, FL presents also different drawbacks. The main challenges to address are: [7] [8]

- Non-IID data: devices produce and collect data in a non independent and identically distributed way among the network nodes.
- Unbalanced data distribution, as also the number of samples across machines may vary significantly.
- Communication: even if data are not transmitted across the network, FL may involve an enormous number of devices, thus communication can be much slower than local computation.
- Systems heterogeneity: as devices are provided with different computational power and communication capacity, it might be difficult to handle such heterogeneity within a federated network.
- Privacy: even if data remain locally, a model update may still expose some private information. Some approaches can alleviate this drawback, but impacting on the performances; thus, it is important to study and analyze federated systems to find a good trade-off between efficiency and privacy.

1.1.1 FL in vehicular networks

In the last years, there has been a great development of the so-called Intelligent Transport System. Many innovations have been made so far, but a lot of challenges are still open.

The issues to face are different, but all of them aim to have a better quality of transportation. This has a lot of implications: more driving safety through hazard detection and trajectory prediction, traffic optimization, lower fuel consumption, decreased traffic congestion, shorter trips time and so on and so forth: for most of them, nowadays, Machine Learning, or rather more specifically, Deep Learning and Reinforcement Learning have been proved to be the state-of-the-art. For this reason, Federated Learning has gain popularity also within the transport domain, as it can be used by vehicular networks.

In this case, besides inheriting the pros and cons explained above, other issues must be taken into account due to the agents' high mobility, which causes frequent and rapid changes in the network connectivity, strongly impacting on the data collection and on the communication side, as devices can face slow connectivity or even total lack of it.

Also, it is important to notice that modern vehicles are provided with an increasingly number of sensors, such as cameras and radar: as already mentioned, they can generate a great quantity of data, which may be difficult to handle, both for computational and storage points of view. Moreover, also energy is a crucial factor: training a ML algorithm requires power, which instead it would be better to use to increase vehicle autonomy. Just this drawback could lead to a low participation in the training phase.

To tackle the problem, a strategy to incentivize users is necessary [4]: for example, in [9] author suggest a method to select users based on their capability, providing appropriate rewards to clients that take part in the learning process. In [10] a "Dynamic Federated Learning-Based Economic Framework for Internet-of-Vehicles" is proposed in order to give the right reward to devices that contribute to the training. The authors address two issues present in vehicular networks: first, due to the mobility of users, the network changes very frequently; second, the so-called Quality-of-Information, QoI, of data collected by vehicles may be very diverse. For this reason, at each round, the coordinator selects only vehicles that are in areas of interest and that have a high value of QoI, which is also used to compute the optimal contracts, necessary to maximize the profits, both of the training devices and the service provider.

A further consideration regarding FL applied in vehicular networks is related to the kind of algorithm to perform: a reasonable and functional approach would imply unsupervised learning or self-supervised algorithms, meaning that labels can be automatically inferred while driving. As a matter of fact, manual annotation of

data is impractical, as the driver is of course unable to do it. Even annotating after driving is not a doable strategy: indeed, the labeling procedure could require a huge amount of time, since the collected data can be many and difficult to annotate. For this reason, computer vision tasks, such as semantic segmentation of surrounding environment, object detection and classification are not indicated for vehicular FL.

This last drawback is also mentioned in [11], a brief report that shows the challenges of FL for vehicular networks, explaining also which can be feasible applications. For the latter point, the authors identify a hierarchical structure for autonomous driving, from the sensing phase to the actuation. Between them, in order, the layers are object detection, identification and tracking, movement prediction and driving decisions. The authors claim that FL would be a good candidate for the higher levels, as labels can be generated by vehicles, in contrast to the first ones, which need accurate ground truth annotations.

1.1.2 Vehicular Knowledge Network - VKN

As already stated, intelligent vehicles have a lot of sensors and cameras, that they use to sense the surrounding environment, generating a massive amount of raw data. The transmissions of such a quantity of information, besides consuming network resources, may results as redundant or not even necessary [12]. For these motivations, a new concept has been introduced: each vehicle should collect data and extract from them a higher knowledge, that can be further exchanged. This allows the creation of a Vehicular Knowledge Network (VKN).

It is worth noticing the difference between information and knowledge: information is referred to as everything that can be sensed through sensors or can be known by sharing data with other nodes of the network; on the other hand, knowledge is something that is extracted from this information, thus has a greater level of abstraction [13]. An example of knowledge is given by a trained machine learning model, characterized by its own parameters. Moreover, knowledge can also refer to what is generated by applying the above-mentioned model, i.e. the output of the model itself.

It is understandable that in such a context, Federated Learning could have a key role in the building of VKNs, as they should be created in such a way to enable and support the development of federated algorithms.

1.2 State-of-the-art

1.2.1 Federate Learning

The vanilla algorithm for FL is the *FedAvg*, proposed in [7] by Google, whose pseudocode is shown in algorithm 1; afterward, in this work, it is also referred to

as the standard Federated Learning method.

At each global step, named round, the central coordinator randomly picks users among the available ones and sends to them the model to train. After that devices perform the local training for a preset number of iterations, named local epochs, they send back the updated parameters to the central coordinator, which aggregates them, simply by computing an average weighted on the number of local samples with respect to the total ones. It has been proved that *FedAvg* is "robust to unbalanced and non-IID data distributions" [7].

Algorithm 1: Federated averaging *FedAvg*

Data: η is the learning rate, n_c is the amount of samples of user c, n_r is the total number of samples of all users picked for round r

Initialization of weights w_0
for *each round* $r = 1,2\ldots$ **do**
  random selection of C clients
  for *each client* c ***in parallel*** **do**
    w_{r+1}^c = Client update(w_r)
  end
  $w_{r+1} = \sum_{c=1}^{C} \frac{n_c}{n_r} w_{r+1}^c$
end

Client update(w)**:**
for *each local epoch* $le = 1,2\ldots$ **do**
  for *each batch* $b = 1,2\ldots$ **do**
    $w = w - \eta \nabla l(w, b)$
  end
end
return w to server

A lot of research works aim to overcome the above-mentioned issues related to FL. For example, in [14], to address systems and statistical heterogeneity in federated networks, authors introduce *FedProx*, which allows to reach higher stability in convergence and better accuracy. *FedProx* shares the principle of working of *FedAvg*, as it can be considered a generalization of the latter. The main differences with respect to *FedAvg* are two. To cope with system heterogeneity, *FedProx* tolerates partial works from users: indeed, it allows different local computational processing, permitting to sent also partial solutions, which are still aggregated by the central coordinator. The rationale behind this concept is that in *FedAvg* all the clients must perform the same amount of epochs, without considering the diverse resources of the devices, such as computational power, connectivity and battery capability. On the other hand, as for statistical heterogeneity, the proposed method introduces

a proximal term in the local function, which must be minimized. This change prevents the updates from being very different with respect to the global model, which is received before starting the training, thus reducing the effect of updates that can negatively influence the convergence.

In [15], the consequences of non-iid data on the training are investigated. With more non-iid data, due to the diverse samples' distributions, the weights computed by clients are more different among each other; because of this, the so-called weight divergence increases. The latter is defined by authors as

$$\frac{||w^{FedAvg} - w^{SGD}||}{||w^{SGD}||} \tag{1.1}$$

where w^{FedAvg} and w^{SGD} are the weights obtained respectively with *FedAvg* and Stochastic Gradient Descent. The difference between distributions can be measured by using the earth mover's distance, EMD: when it increases, the test accuracy decreases. To overcome this drawback, the authors propose a solution: basically, as there is "no control on the clients' data", they "distribute a small subset of global data containing a uniform distribution over classes from the cloud to the clients". By increasing the amount of distributed samples, also the test accuracy grows, while the EMD gets smaller.

In [16] the communication cost issue is faced, with the proposal of two ways to reduce the uplink transmission load. The explained methods are structured updates, where the weights are learnt "from a restricted space parametrized using a smaller number of variables", and sketched updates, in which the model update is compressed before the transmission to the central coordinator. Such techniques may still be useful, because there might be a communication bottleneck, due a huge number of participating devices in FL training and due to the size of the model to exchange.

As trained model parameters may still expose some private information, in [17], authors introduce differential privacy, which may be applied also in FL to increase data security. Other used techniques related to privacy and security are Secure Multi-party Computation (SMC) and encryption [5]. The purpose of Differential Privacy, DP, is to prevent "the FL server from identifying the owner of a local update". Basically, it "adds a certain degree of noise in the original local update while providing theoretical guarantees on the model quality" [4]. On the other hand, Secure multi-party computation [18], also simply multi-party computation (MPC), is a subfield of cryptography. SMPC is a protocol that allows distributed computation over multiple users, which keep their own data private, without sharing them among each other.

It is important to notice that as concerns the aggregation made by the central coordinator, there are two classes of FL, synchronous and asynchronous. In the first case, which is the most common approach, parameters aggregation occurs only

at the end of every round, after all participants complete the local training; on the contrary, the second method relaxes this constraint. However, asynchronous strategies do not provide convergence guarantees [4] and require an enormous number of updates [7], reasons for which most of the studies focus on synchronous FL. For the same motivation, also in this work asynchronous FL is not taken into account.

Frameworks and datasets

So far, different open source frameworks have been developed for Federated Learning. The main ones are TensorFlow Federated (TFF) [19], PySyft [20], LEAF [21] and FedML [22].

TFF is an open-source framework developed by Google, which has used FL to train a next-word prediction model for mobile keyboards [23]. TFF furnishes datasets and models for FL and provides a flexible interface for further studies and development.

PySyft is a Python library for FL, which includes tools for security and privacy such as Differential Privacy, and Encrypted Computation methods, like Multi-Party Computation (MPC) and Homomorphic Encryption. Recently, also the support for TensorFlow has been added. As concerns homomorphic encryption [24], it permits to perform computations directly on encrypted data, without needing to decrypt them first. After the processing, the output is still encrypted: its decrypted form is equal to the result obtained by computing on not crypted data.

LEAF is another open-source framework [21], whose main contribution is providing datasets build specifically for FL. To date, six datasets are present in LEAF:

- Federated Extended MNIST, FEMNIST, in which data of EMNIST [25], consisting of handwritten digits and letters, are divided by writer; characters are centered in grey-scale images of 28x28 pixels;
- Sentiment140 [26], which collects tweets from different users in order to build a sentiment analysis dataset;
- Shakespeare, that includes the works of the famous writer, taken from [27] : here the different users are the roles from each play;
- CelebA [28], which is a collection of face images and attributes of celebrities, of course partitioned based on the represented person;
- Reddit, which gathers comments published on the platform on December 2017, dividing posts by authors;
- a synthetic dataset, built specifically for FL. For more details refer to [21].

Another available library for Federated Learning is FedML. The latter allows to perform not only simulations on a single machine, but it permits distributed computing and to train on different kinds of IoT devices and smartphones as well. Moreover, it provides useful benchmarks for FL and also the possibility to implement algorithms of vertical FL and split learning.

As concerns Vertical FL, it is a different paradigm with respect to the horizontal one, which has been implicitly considered in this work. In horizontal FL, users' data belong to only one features space and clients are unique; in vertical FL, instead, users of distinct datasets are the same and each set is characterized by different features [5]. For instance, horizontal FL may be exploited by different hospitals by using the same medical tests prescribed to different people. On the other hand, Vertical FL can be utilized by hospitals that do different medical tests, but to the same group of patients.

Split learning is another paradigm used in FL: "in this setting each client, (for example, radiology center) trains a partial deep network up to a specific layer known as the cut layer" [29]. In [30], the authors compare *FedAvg* with this method, named *SpliNN*, finding that the latter needs much less computation, measured in TFLOPS, and also fewer transmitted data.

1.2.2 Gossip Learning

Gossip learning is an alternative approach to FL: it also relies on data remaining at the edge devices, but without the need of a central controller for model aggregation. The basic implementation involves that each participating node should first initialize a local model; then, when it receives a model from another user, it updates its local one, either simply overwriting it, or aggregating both of them, following different averaging strategies.

In [31], authors find that gossip learning, under a uniform distribution of data among the nodes, may outperform federated learning. In the other cases, its performances are actually similar to standard FL. As stated in [31], "the advantages of gossip learning are obvious: since no infrastructure is required, and there is no single point of failure, gossip learning enjoys a significantly cheaper scalability and better robustness."

Besides this, from the literature review it seems that gossiping learning is not analyzed as much Federated Learning.

A drawback of Gossip Learning is that its performances are strongly dependent on the network topology, as stated in [32]. For example, in the case of vehicular networks, data needed to train a model may be possessed by only a few agents, which can also be geographically sparse, reducing the probability that they will be close enough to directly communicate with each other. In such a case, there would be the necessity to connect them by using an extensive infrastructure, but, with

this premise, Federated Learning can be more indicated. Also, even if agents form local clusters that can not communicate with each other, they will end up building specific models that may be not good for the global function optimization.

In this work, a novel method is proposed, named Hybrid Federated Learning, as in can be viewed as a combination of Gossip and Federated Learning: it leverages the possibility to exploit locally a mechanism like gossiping, but allowing, thanks to the global aggregation, to alleviate issues that may arise due to topology of the network.

1.2.3 Hierarchical FL

Even if a few works can be found in literature, another approach that can be related to the introduced method is Hierarchical Federated Learning. The basics behind this strategy are to create a stratified structure of devices, performing model aggregation also in intermediate layers, placed between users and central coordinator, in a hierarchical manner. The latter factor is the point that distinguishes Hierarchical FL from Hybrid, in which merges are only computed by the server and are also less frequent, as devices receiving a model from another user do not merge it with the local one, that is instead simply overwritten with the obtained model, becoming the new starting point for training.

In [33], authors underline the limits of the existing FL systems, which usually operate within a wide-area network, WAN, as devices can be very widespread, covering large geographical areas. Indeed, WAN is usually slow and constrained, causing a communication bottleneck; second, it is also very expensive. Thus, they propose a hierarchical aggregation mechanism, called *LanFL*, based on structures relying on the local-area networks (LANs), as they provide a huge quantity of bandwidth resources, besides having a cost that is way less with respect to the WAN. *LanFL* has been proven to decrease training time and reduce WAN traffic. Indeed, by applying this framework, global aggregations are much less frequent; on the contrary, local merges, which are performed by some devices within the LAN, are very copious.

This approach is also studied in [34], where the authors introduce a "client-edge-cloud hierarchical Federated Learning system". Between cloud server and users, also edge servers are exploited. The latter, after a certain amount of local updates, perform the models aggregation from their own clients. Afterwards, when a definite number of edges aggregations is computed, the cloud server merges all the edges' parameters. By using this architecture, less training time is needed compared to standard FL.

Particularly, in [35], the hierarchical method is applied in mobile networks. Depending on their position, mobile devices are grouped and assigned to a base station, BS, to which clients send the computed gradients, instead of transmitting

them to the central server. The BS average the gradients and the obtained result is sent back to its associated units. Also in this works, the intra-cluster aggregation is coupled with inter-cluster model averaging, performed by the coordinator; this is a crucial step, without which it would be very unlikely to converge to a global model that can be generally valid, regardless of the BS. The main advantages of the proposed framework are two: it achieves higher accuracy, and, as this technique necessitates shorter transmission ranges, communication latency is lower.

1.3 Contributions

The contributions of this work are the following ones. In chapter 2, it is analyzed how the different simulation settings influence the federated learning, applying *FedAvg*, the vanilla algorithm. Then, a novel approach, called Hybrid Federated Learning, HFL, is introduced. Experiments with FEMNIST and CIFAR-10 datasets are simulated, to compare and explain the advantages of the presented method with respect to *FedAvg*. In chapter 3, both approaches are implemented with NGSIM US-101 dataset, with the aim of training a trajectory prediction model. The main objective of this simulation is to prove the feasibility of FL, and more specifically of HFL, within vehicular networks.Finally, chapter 4 contains the conclusions; ideas and steps for future work are presented as well.

Chapter 2

Hybrid FL

In the context of Federated Learning, a lot of works focus on reducing communication costs. A natural approach consists in reducing the number of global rounds, while increasing local computation, but this could lead to worse performance and delayed convergence [4], as also shown in some experiments of [7] with non-iid data. For iid case, instead, increasing local computations lead to improvements, but it is important to recall that generally data are non-iid.

The rationale behind this downside is well explained in [14], where authors claim that when users have very diverse local objectives (this occurs due to their sample distributions), a greater amount of local steps may lead each device to only optimize its own objective function, with the possibility of negative effects on the global one. Thus, it is important to find a trade-off between global rounds and local epochs. Other techniques to limit communication costs are the ones cited before, i.e. structured and sketched updates, explained in [16], but they can cause issues in accuracy and convergence.

The proposed approach, called Hybrid Federated Learning, is orthogonal to these works and its main benefit is reducing the number of transmissions needed for training.

As said, this method can be considered as hybrid between Federated Learning and Gossip Learning. Indeed, it still relies on a central controller that aggregates weights, by using a weighted average as in *FedAvg*. Its peculiarity consists in gathering users in groups of n members, such that only one of them receives the updated model from the controller.

2.1 Algorithm design

Figure 2.1 shows the schemes representing the principles of Federated Learning and the proposed approach, whose pseudocode is shown in algorithm 2. The first step is

the initialization of the weights; in [7] it has been proven that a shared initialization, meaning that all starting weights are equal among all users, is beneficial for FL.

For each round, a number C of clients is picked for the training and are grouped together in G clusters of size N. Only one member per group receives the latest available model from the central coordinator; then, after the local training, instead of sending back the learnt parameters to the server, it transmits them to another user of the ensemble. At this point, this sequential process continues until the last remaining member of the cluster, after the training, finally sends the obtained model to the controller. This mechanism is executed in parallel by all groups.

When the server receives from each cluster the updated parameters, it computes the weighted average based on the total samples of each ensemble; after this stage, a new round starts, performing the same procedure described above.

2.1.1 Grouping strategy

Regarding the strategy for grouping clients, in pseudocode 2 no specific rules are present. It is important though to mention the criteria according to which ensembles may be built, as this is a crucial step for HFL.

The simplest technique is creating clusters randomly, as done in the experiments based on the FEMNIST and CIFAR-10 datasets. A further strategy consists in gathering clients according to the kind of samples they have; this approach is applied in section 2.6.3, in which the simulations are executed using the CIFAR with non-iid data.

However, in a real case scenario, these methods have a great limitation, since it must be taken into account the location of the users in order to exploit direct transmission of the model. Therefore, it is necessary to gather users that are geographically close to each other. When the position constraints are satisfied, it may be possible to form clusters in a random manner or also considering the type of data of each client.

2.2 Context and use case

Hybrid FL can be exploited in different scenarios. For example, consider a rare event such as a car accident: some vehicles can collect data about it, but due their mobility, they do not have much time and can gather a limited amount of information. In such a case, a vehicle can use these data to train a model for accident prediction; then, it sends the updated parameters to a vehicle next to it that has stored data about the same accident, maybe even also from a different perspective. This process is chain repeated in the area around the accident, until the last member of the local cluster transmits the model to the central server,

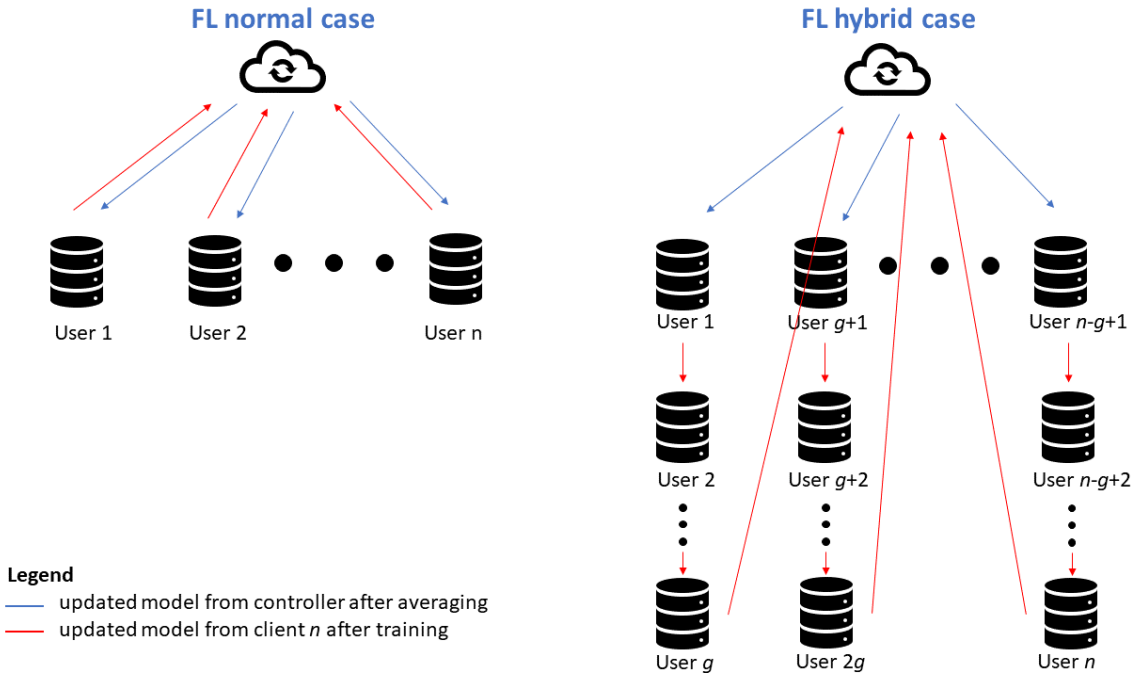


Figure 2.1: Scheme representing standard FL and Hybrid FL

which can merge models coming from different and geographically spread groups that have collected data of different accidents.

Another situation in which this method can be used is for trajectory forecasting. By using their sensors, vehicles can continuously detect objects and estimate their position: with this assumption, it is desired to train a model able to predict the future location of agents considering their past states. For instance, if vehicles are driven on highways, they can form groups based on their position, sequentially exchanging model updates.

On the other side, if one wants to train a general model, valid not only in highways, since they have a specific mobility pattern, it can be possible to form ensembles which, besides considering the position, can be built taking into account other factors, such as the kind of data collected by vehicles. In this way, it would be feasible to peak agents having different data, such that the model related to each group would be updated by using heterogeneous inputs, thus limiting the risk of sending parameters biased due to specific samples distribution.

Algorithm 2: *Hybrid Federated Learning*

Data: η is the learning rate, n_g is the amount of samples of group g, n_r is the total number of samples of all users picked for round r

Initialization of weights w_0
for *each round* $r = 1,2\ldots$ **do**
  random selection of C clients
  grouping C clients in G groups of N members
  for *each group* g ***in parallel*** **do**
    w_{r+1}^g = Group update(w_r)
  end
  $w_{r+1} = \sum_{g=1}^{G} \frac{n_g}{n_r} w_{r+1}^g$
end

Group update(w)**:**
for *each client* $c \in g$ **do**
  w = Client update(w)
end
return w to server

Client update(w)**:**
for *each local epoch* $le = 1,2\ldots$ **do**
  for *each batch* $b = 1,2\ldots$ **do**
    $w = w - \eta \nabla l(w, b)$
  end
end
return w

2.3 FEMNIST simulations

For this works, PyTorch framework is used, due to its simplicity and since it allows great freedom to perform the intended experiment. The used dataset is the forementioned FEMNIST, in which only digits are considered to limit the computational load.

At first, experiments are run using it, and CIFAR in the next section, even if they are not related to the transportation domain and even though FL is not indicated for supervised learning that needs external sample annotations. Nevertheless, these two data collections are easy to use, allowing to draw some first conclusions for the comparison between the standard and hybrid strategy.

For the loss, the *Cross Entropy Loss* is utilized, while the used optimizer is the

Stochastic Gradient Descent, with a learning rate equal to 0.01 and momentum of 0.9. The batch size is set to 8, the number of total rounds is 50 and, finally, training involves only one local epoch, if not specified otherwise.

The used model architecture, shown in figure 2.2, is simple Convolutional Neural Network, which is equal to the one present in [36]. It is composed of two convolutional layers, each of which is followed by max-pooling and ReLU, Rectified Linear Unit, activation function. Then, the resulting output is passed to a first fully connected layer, followed by ReLU and a dropout layer. Finally, there is the last fully connected liner layer, whose output is given to a logarithmic softmax function; the index of the maximum returned value is the predicted class of the input data.

It is worth noticing that in all experiments test accuracy is computed using a test set, which has no data in common with users' data, by the central coordinator. This choice has been made just for simplicity, but depending on the use case the testing phase can be also made by the clients themselves.

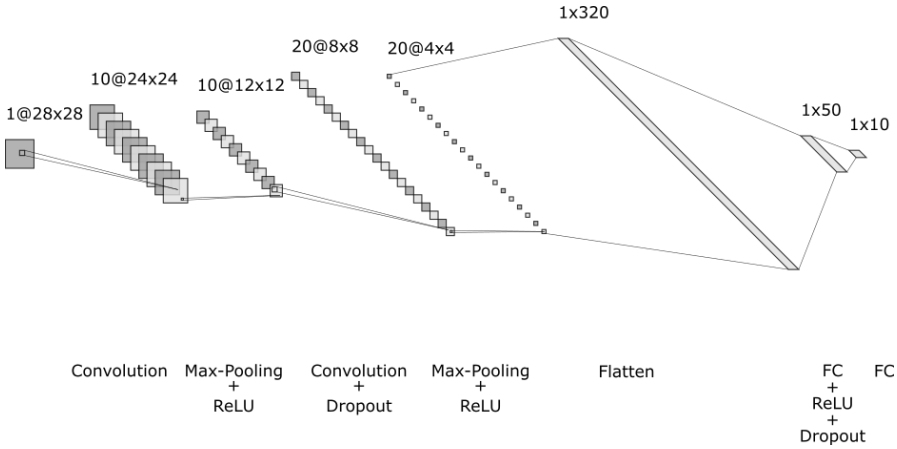


Figure 2.2: CNN used for FEMNIST dataset

FEMNIST statistics

Since the original FEMNIST dataset is prepared in order to remove letters, in this section some details about the final processed dataset are provided. Figure 2.3 shows the users statistics, in particular the number of clients with respect to the number of samples. As it can be noticed, most of the clients have between 100 and 120 data; just a few users own less than 50 samples. Other relevant information is given in table 2.1. On the other hand, figure 2.4 shows the distribution of all samples with respect to the class they belong to: as it is possible to see, the number of occurrences is quite similar among the digits, meaning that the dataset can be considered balanced from this point of view.

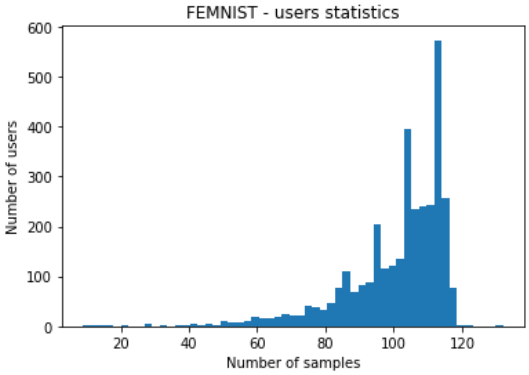


Figure 2.3: FEMNIST without letters - users statistics

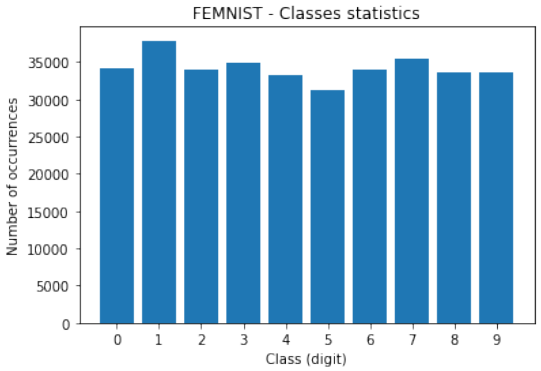


Figure 2.4: FEMNIST without letters - classes statistics

Total number of users	3383
Total number of samples	341873
Mean (samples per user)	101.06
Standard deviation	14.72

Table 2.1: Statistics of FEMNIST without letters

2.3.1 Preliminary experiments

First of all, some simulations are performed in order to understand the influence of some parameters on the training phase.

Figure 2.5 shows the test accuracy with respect to a different number of users, which are randomly selected at each round. Apart from the case with only one client, the trend is the same, but increasing the users the curves become smoother, presenting much less pronounced oscillations.

Then, the consequence of the splitting of the data is analyzed. To do this, only the first 150 users are considered, simulating different scenarios. Indeed, users are gathered in group of x members, in order to put together data from different clients, simulating fewer devices with more samples, but keeping the overall amount of data constant. As shown in figure 2.6, training speeds up and reaches better accuracy when having fewer users, but with more inputs.

However, gathering samples from users is not doable in Federated Learning: the Hybrid strategy tries to alleviate this limitation, since clients of a group, instead of transmitting data, share updated weights among each other, so that the model sent to the coordinator has been trained using more samples.

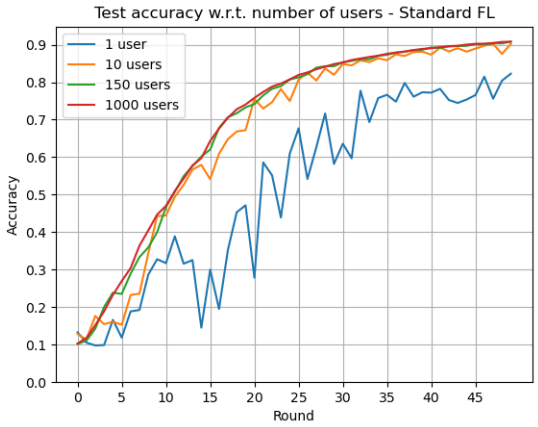


Figure 2.5: FEMNIST test accuracy w.r.t. number of clients in standard FL with random selection of clients

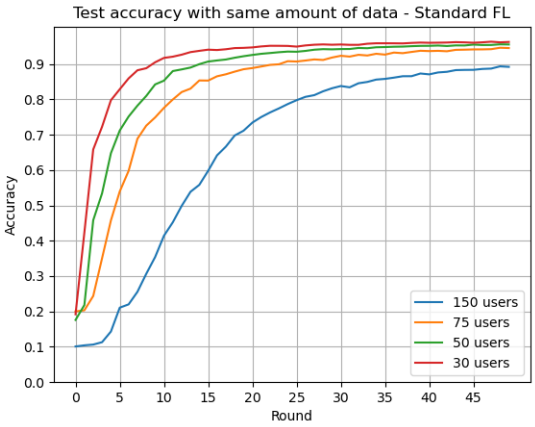


Figure 2.6: FEMNIST test accuracy with same amount of samples, using standard FL with data of first 150 users

2.3.2 Hybrid FL experiments

Figure 2.7 highlights the comparison between the standard approach, i.e. *FedAvg*, and the Hybrid FL, using as before only the first 150 users: the plot shows that grouping clients in ensembles of 2, the test accuracy grows faster and reaches a higher score, while with groups of 5 agents performances are even better.

Due to the apparent benefits of HFL, other simulations are executed, without the limitation of using only the first 150 users. Figures 2.8 and 2.9 show a more fair comparison between the standard approach with respect to the proposed one. In this case, in each round, 150 users are picked randomly for the training; as it is possible to see, the hybrid versions, under the same number of epochs, leads to faster convergence and higher accuracy. Moreover, the plot shows that also in this experiment, with the same number of epochs, grouping users in ensembles of 5 is better than having groups of 2 members. Also, both for SFL and HFL considering the same group size, performances get better when increasing the number of local epochs.

It is important to notice that in the simulation with only one local epoch and groups of 2, performances get close to the standard case, but with two local epochs, i.e. the double of the computation; then, the hybrid case with one epoch and groups of 5 members largely outperforms the standard case with 2 epochs, while it is comparable with the standard approach having 5 epochs for each device.

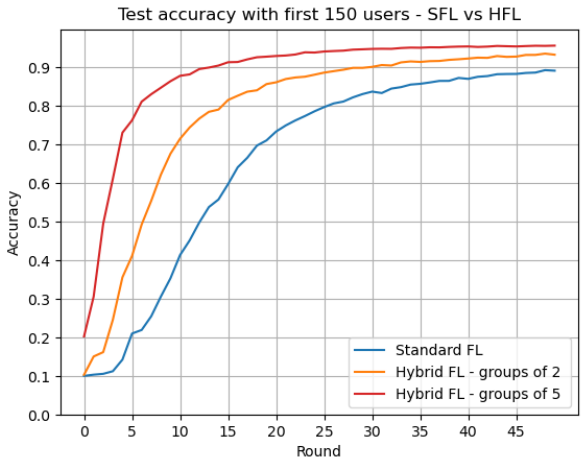


Figure 2.7: FEMNIST test accuracy - comparison between SFL and HFL, using first 150 users

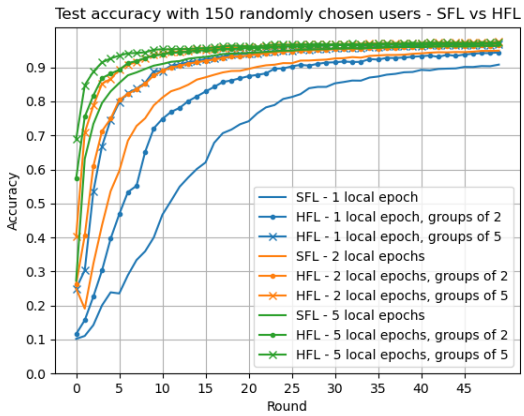


Figure 2.8: FEMNIST test accuracy - comparison between SFL and HFL

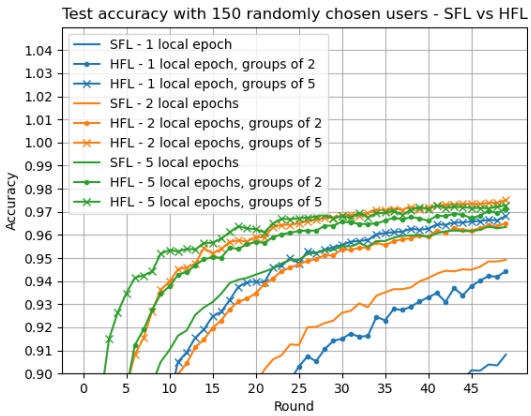


Figure 2.9: FEMNIST test accuracy - comparison between SFL and HFL (focus on accuracy higher than 0.9)

2.3.3 Non-IID data

So far, data are divided by authors: this means that data are non-IID, as each writer has its own calligraphy.

However, even under these assumptions, handwritten digits are similar among people; for this reason, another experiment is executed. Data are divided into two categories, depending whether the number is greater than 4. Basically, each user has samples belonging to only half of the total available classes; from now on, in

the case of digits FEMNIST dataset, the non-iid configuration refers to this specific extreme setting and not on the original one. With this design, the learning process is much worse than previously, as it is clear from the blue curve in figure 2.10. On the other hand, the orange and green curves show the trend by applying the hybrid approach, in which users are gathered in groups of two. The difference between these last two cases concerns the user getting the updated model from the other group member. In both cases, the receiving users are picked randomly, but for the orange curve it is chosen in such a way it has data belonging to the classes not present in the first client's dataset; for the green line, instead, this selection is made purely randomly, without this constraint.

It is interesting to note that in the last case the accuracy grows faster, but in the end both the green and the orange curves converge to the same values. However, as concerns the different behaviour during the first rounds, it is worth mention that the trend can depend on the chosen seed and that, in any case, users are just divided into two classes, making difficult coming to a valid conclusion about the strategy to build groups. Anyway, also in this scenario, the introduced approach brings benefits to the training, as, under the same number of local epochs, the hybrid strategy outperforms the standard one.

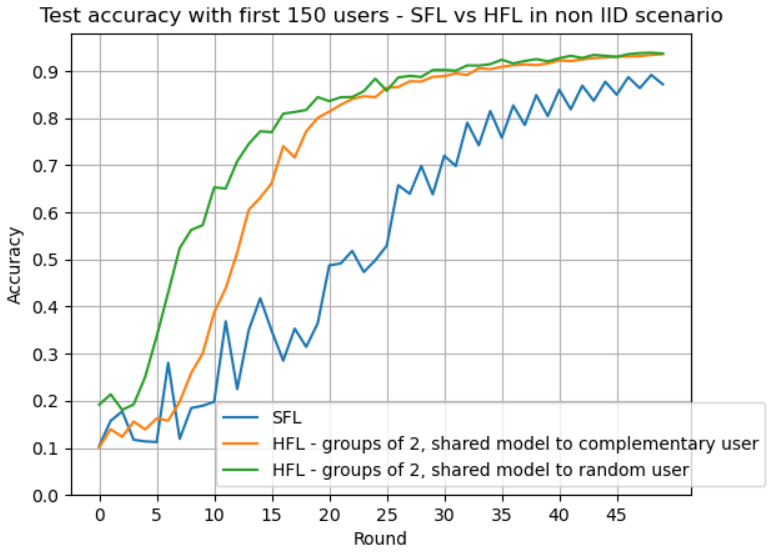


Figure 2.10: FEMNIST test accuracy - comparison of test accuracy between SFL and HFL, with data of first 150 users distributed in a non IID configuration

2.3.4 Advantages of proposed methodology

The proposed strategy presents some main advantages: first, with equal rounds, accuracy is higher with respect to the standard method. This also means that the accuracy improves more rapidly, allowing to reduce the number of rounds to achieve the same performances, with a consequent lower amount of communication costs. Moreover, under the same global rounds and assuming communication among vehicles is exploited, the number of transmissions is smaller, due to the fact that at each round the central coordinator does not send and receive the model from every node involved in the training phase.

Given C users and groups of N members, in the standard approach, at each round there are $2C$ transmissions, while this quantity is equal to $C+\frac{C}{N}$ in the hybrid version. Indeed, the server sends the updated parameters only to one member for each group, which are $\frac{C}{N}$, while the C transmissions are due to the users: all of them, during a round, send the model only once, either to another group's member or to the server. Figure 2.11 shows the number of transmissions per round as function of the number of users and group dimension, which can be considered equal to one in the case of SFL.

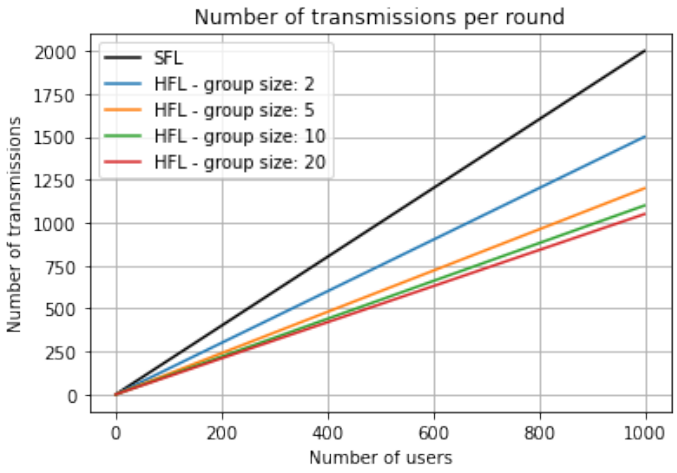


Figure 2.11: Number of transmissions per round depending on the number of users and group size

Last but not least, this technique may exploit communication among vehicles, allowing to relax the load on the servers. The advantages from the communication point of view are better analyzed in section 2.5.

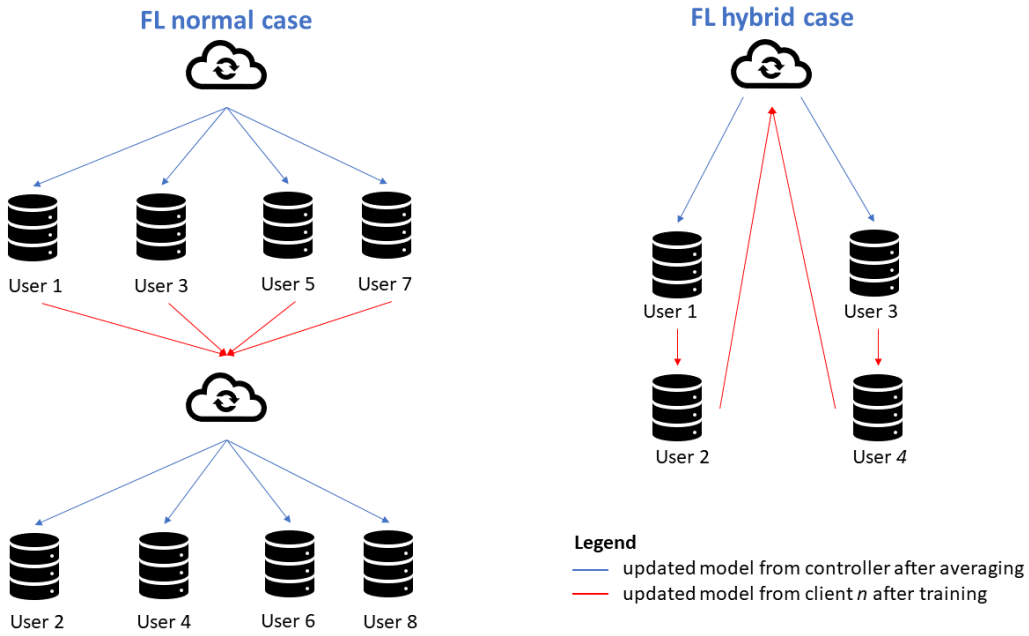


Figure 2.12: Scheme for comparison between standard and hybrid FL, with 4 users per round

2.4 Visual explanation in i.i.d. settings

A possible reason that might explain the improvements of the hybrid approach is explained in this section.

FedAvg introduces a great parallelism, as all the users picked for each round work at the same time; on the other side, the Hybrid approach breaks this parallelism: indeed, within a group, training is serialized across members, which have to wait for the updated parameters from another client. The point is that in i.i.d. scenario, users may compute roughly the same computations, meaning that updates coming from different devices will be similar, thus sending to the central coordinator redundant information. A proof that supports this hypothesis is given in figure 2.5, which highlights that the trend of test accuracy is mostly the same in the case of 10, 150 or 1000 users.

On the left of figure 2.12, the standard case is represented: the users of the first round perform their training, after which they send the model to the server, which averages them; it may happen, especially in i.i.d. condition, that the clients' models are quite similar among each other, thus to the average model itself. For the second round, the other users continue the training, after receiving the global updated weights. On the right, it is represented the hybrid case: users 2 and 4 receive the model directly from user 1 and 3, respectively. Since these models are

comparable to the average one of the standard case, the computations are quite the same with respect to users 2 and 4 of the normal version, but only one round is needed, instead of two.

In non-iid settings, models of 1 and 3 may be not comparable and can be more diverging. However, users 2 and 4 begin the training starting from updated models, whose parameters are better for the global objective function with respect to the ones that devices 1 and 3 received by the controller. Again, in the standard approach this occurs in the second round, while in the hybrid still in the first one. Because of that, also in this scenario hybrid approach can help to speed up the training.

In other words, it is possible to analyze this concept from the coordinator point of view: in the standard case, the server receives a lot of models, each of which is the output of the local training, performed only by using the dataset of the relative device; on the other hand, in the hybrid strategy, the server obtains fewer models, that are instead the result of a training executed on a higher quantity of data, as they depend on the sets of more users. This means that every update sent to the server, as more samples are utilized to get it, carries larger improvements towards the global optima.

2.4.1 Linear Regression with SGD

To visually explain the concept with iid data, it can be useful to consider a simple case of linear regression using Stochastic Gradient Descent, SGD. In the example, there are four different users; each of them has a dataset consisting of 100 points, which have the x-coordinates in the range [0,2). For each sample k, the y-coordinate is computed by using the following equation: $y_k = 4 + 3 * x_k + \epsilon$, with ϵ random variable such that $\epsilon \sim U[-1,1]$. Dataset of the first user is represented in figure 2.13; the other ones are similar to it, indeed in this case data are distributed in an i.i.d. manner. The objective is to perform linear regression through SGD in a federated way, setting the learning rate equal to 0.01. The standard case, in which updated weights are averaged after each round, is compared to the proposed hybrid algorithm as represented by figure 2.14. Specifically, the results of iterations 1, 10 and 50 are shown. At the upper-left angle of each plot, it is possible to see the points of the different datasets, which are colored according to the user to which they belong. However, figures relative to the first iteration do not show them, as the regression line is still far from them.

As it is possible to notice, in the standard case all the lines are mostly overlapping, thus also their mean value. This basically means that all of them are computing

broadly the same updates. In the hybrid approach, users 1 and 3 perform their local training and then share the model respectively with users 2 and 4. Client 1 and 3 make almost the same computation, as proven by the fact that their regression lines are similar among each other; moreover, these curves are also quite the same as the ones of the standard case (of course considering the same iteration). On the other hand, users 2 and 4 perform the regression starting from the weights received from 1 and 3, avoiding to make computations that would have led to parameters comparable to the obtained ones, with the consequence that the final average line is closer to the right solution with respect to the normal case one.

From the plots, it is clear that the hybrid approach is faster with respect to the number of rounds. Note that at round 50 curves are overlapping also in HFL, but just due to the zoom level.

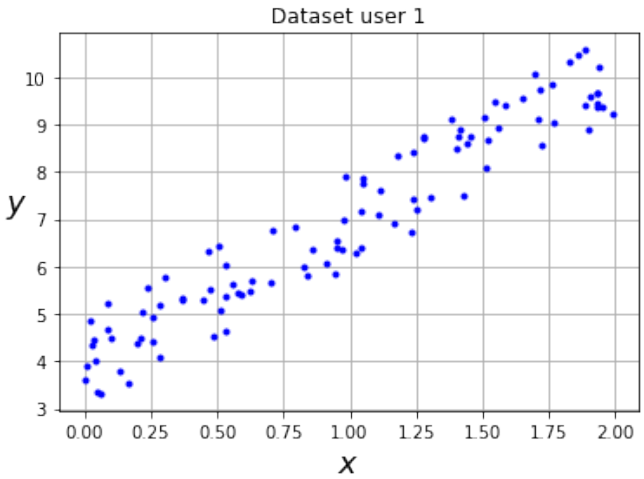


Figure 2.13: Linear regression by using SGD: dataset of user 1

2.5 Beyond rounds: time and number of transmissions

Until now, all the plots have been made only considering the rounds: this metric takes into account the number of local computations, as during it, both in SFL and HFL strategy, the number of devices that train the model is equal to the number of picked clients. Nevertheless, other factors have to be taken into account, such as the time and the number of transmissions.

As already said, the hybrid approach breaks the parallelism among all users, introducing a serialization within the clusters. This has a consequence on the time aspect, as some users must wait for receiving the updated model from a specific

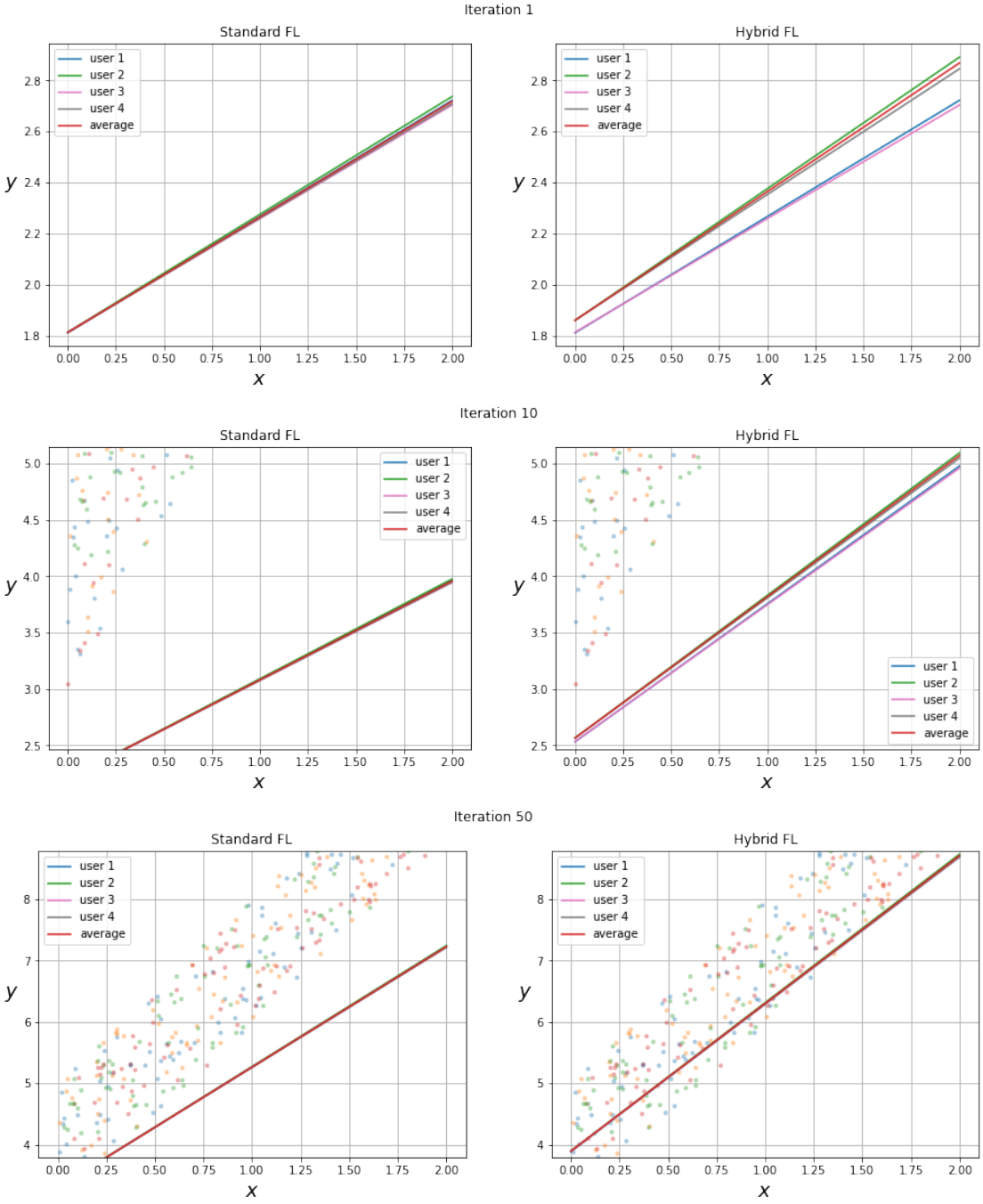


Figure 2.14: Linear regression for standard and hybrid FL over iterations

client, which should wait for another user itself. This depends of course on the number of members per groups.

Before comparing the different experiments, it is important to mention that time is expressed in terms of slots: the time slot is intended as the time needed for computation in a round of *FedAvg*; on the other hand, in the hybrid approach a round takes as many slots as the number of members per group. For now, the time necessary for the model transmission is not taken into account. This choice is not made just for simplicity: indeed, in this work, due to utilised data, a very simple CNN is used, thus it would be useless in a real case scenario. Therefore, it would be necessary to have in mind the use case, the size of the neural network architecture and, finally, the type of infrastructure for the communication and the relative transmission rate.

Figure 2.15 contains different plots that show the test accuracy of *FedAvg* and the hybrid case with respect to rounds, time and number of transmissions. Each plot of figure 2.15 presents three curves:

- standard FL, in which 150 users are picked at each global round;
- standard FL, with 75 clients selected at every round;
- hybrid FL, in which 150 users are picked per global round; they are randomly grouped in clusters of two members.

Observing at figure 2.12, one could claim that assuming an equal number of users per round, actually at the same time fewer clients are utilized in the hybrid case. This is exactly the rationale behind the reason for which also the normal case having half of the users, 75, is taken into account for the comparison. In the latter case, the number of rounds is 100, such that the time needed to finish the process is the same as the hybrid version.

Figures 2.15a, 2.15c and 2.15e are referred to the standard FEMNIST dataset: as regards the rounds, the hybrid approach outperforms the standard one, for which there are no differences depending on the number of clients chosen at each round, as already found in plot 2.5.

With respect to time slots, all the three approaches are comparable, but note that the SFL case with 150 users lasts half of time than HFL: in the latter case, due to the serialization, a round takes more time, thus with the same number of global steps the HFL overall training is longer; however, here the focus is on the test accuracy and, considering the same time interval, its trend is similar to the SFL one.

For the number of transmissions[1], instead, the hybrid method still outperforms

[1]Recall that the number of transmission is equal to $2C$ for the normal approach, while it is $C + \frac{C}{N}$ for the hybrid strategy, where C and N are respectively the total number of users picked at each round and the number of members per group.

the other two, even if the difference with the normal case having 75 users is less noticeable.

As regards the simulations with the dramatic case of non-iid data, whose results are shown in figure 2.15b, 2.15d and 2.15f, the same conclusions can be drawn, but apart from them, here there is another advantage: indeed, curves based on normal FL are very fluctuating, while they are smoother with HFL.

Comparing the hybrid strategy with the standard one having half of users is fair if one takes into account the number of devices training in the same time, but actually it is not when considering the amount of data used within a round, since the aggregated model is derived from fewer users. This has limited implications with iid or not extreme non-iid data, as shown in figure 2.15e, but with dramatic non-iid setting using fewer samples in a global round could be not a good strategy, because this can lead to evident oscillations, as in plot 2.15f.

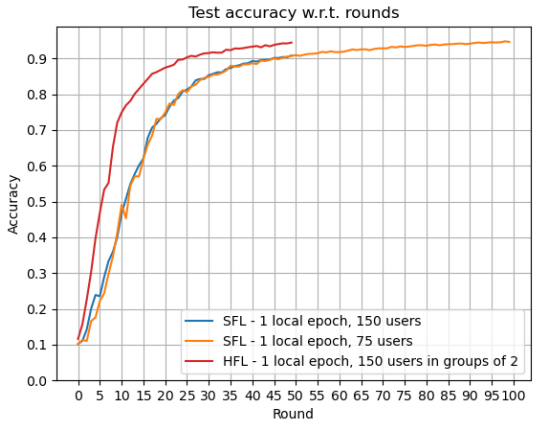
The negative effects of using fewer data are also experimentally proved in section 2.6.3.

2.5.1 Communication

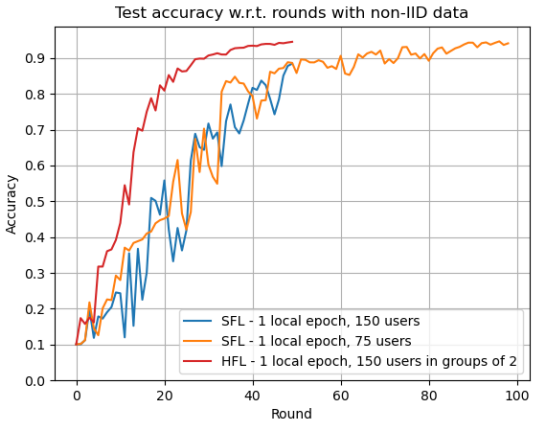
From a communication perspective, the benefit that the hybrid method brings does not concern only the number of transmissions, but there is one important factor not pointed out by the aforementioned plots: indeed, as devices transmit the model's parameters to other ones, it is not necessary to exploit communication channels that link users to the central coordinator. On the contrary, it is possible to make use of device-to-device communications, relaxing the load on the server. This is a great advantage, as also mentioned in [33]. Recall that in this paper, the authors claim that a FL design relying on a wide-area network, WAN, leads to high communication costs and slower model convergence, since WAN may be constrained and unstable, making the network a great bottleneck. For this reason, they propose a hierarchical aggregation design based on local-area network, LAN, that has the main advantages of greater bandwidth resources, which, moreover, are cheaper to exploit. This work refers to a model trained by mobile phones; however, with the right infrastructure, the same conclusions may be drawn also for vehicles, thus can be extended to the current study, too.

2.6 CIFAR-10

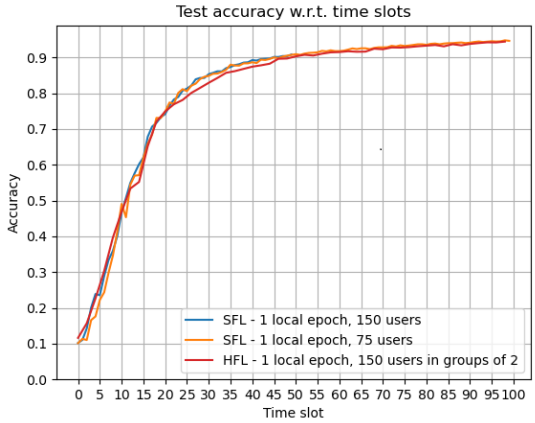
Until now, for the different experiments, the FEMNIST dataset, in which only digits are considered, has been used. To extend the validity of the proposed approach, also the CIFAR-10 dataset [37] is utilized. The latter consists of 60000 RGB figures, having a resolution of 32x32 pixels. There are respectively 50000 and 10000 training and test images, which belong to 10 different and mutually exclusive classes.



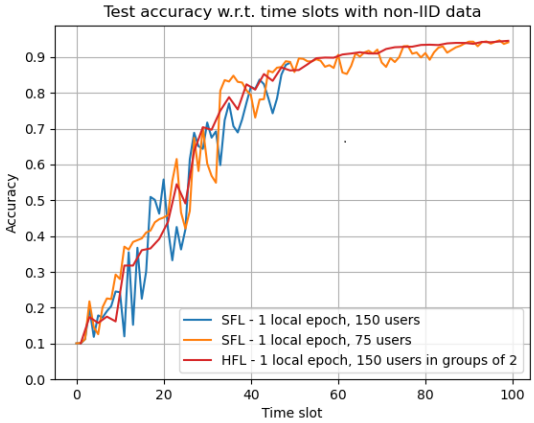
(a) Native FEMNIST w.r.t. rounds



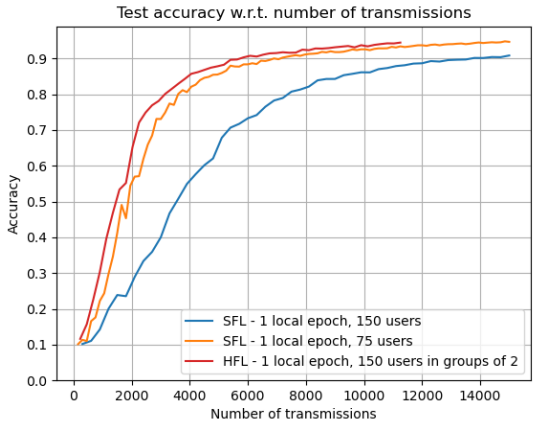
(b) Extreme non-iid data w.r.t. rounds



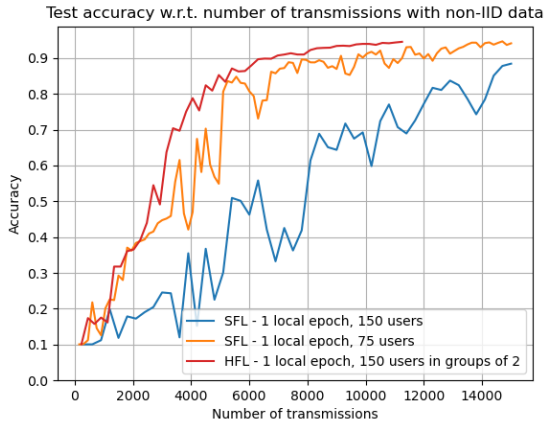
(c) Native FEMNIST w.r.t. time slots



(d) Extreme non-iid data w.r.t. time slots



(e) Native FEMNIST w.r.t. transmissions



(f) Extreme non-iid data w.r.t. transmissions

Figure 2.15: FEMNIST test accuracy - comparison between SFL and HFL in different settings

The reasons that led to choose this dataset are that figures are small, allowing a relatively fast training phase. Nonetheless, there is a great difference with respect to MNIST dataset, another point that has pushed to use CIFAR. Indeed, in MNIST images, due to the pre-processing, the characters, regardless of the class, are centered and surrounded by a black background; thus, every client learns that the sides do not contain relevant information for digit recognition.

On the other side, CIFAR 10 figures, even if they belong to the same category, are very different from each other and have variegated details, such as the background, the colors of the subject and of the surrounding environment, the angle from which the figures are represented. An evidence of it is that, within the same category, images can be further divided into sub-classes. For example, in the class "ship", it is possible to see pictures of little boats, cruise ships and even merchantmen.

2.6.1 Experiments design

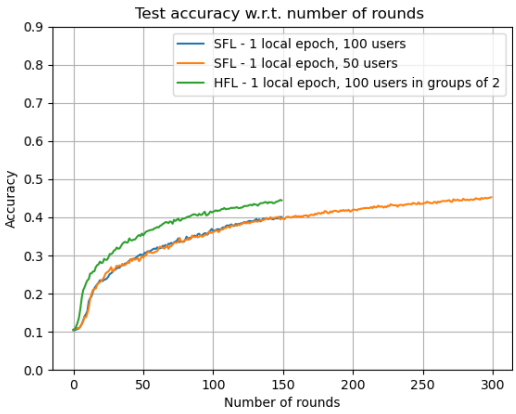
Also in this case, a CNN is used; the model architecture is the same as the previous experiments, with just some adjustments to support slightly bigger figures and the presence of 3 input channels, instead of only 1.

As concern the general design, the test is still done by the central server, while for the setting of the training datasets the details are given together with the explanation of each experiment. For the cost function, the *Cross Entropy Loss* is used, while the utilized optimizer is the Stochastic Gradient Descent, with momentum of 0.9 and learning rate equal to 0.01 and 0.001, respectively for iid and non-iid data. The batch size is set to 16 and training involves only one local epoch.

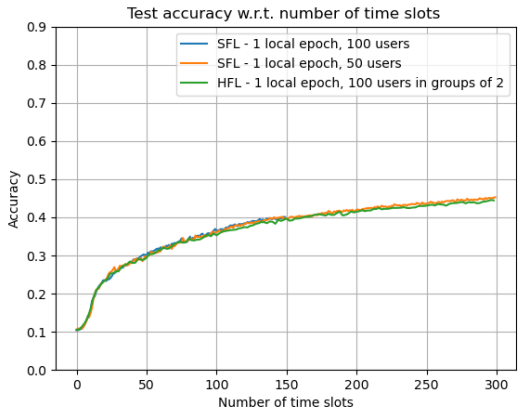
2.6.2 IID data

The first experiments are run splitting the data among the users in an iid way. More specifically, the total number of clients is set to 500, each of which has 100 samples. Samples are assigned in a random way, meaning that they are independent and identically distributed. As in section 2.5, also in this occurrence, 3 cases are considered: two of the are referred to the standard algorithm, while the last one is hybrid approach; the number of clients selected at each round is respectively 100, 50 and 100. Results of this simulation are represented in figures 2.16a, 2.16b and 2.16c, which display the test accuracy trend with respect to the three metrics mentioned before: the number of rounds, time slots and transmissions.

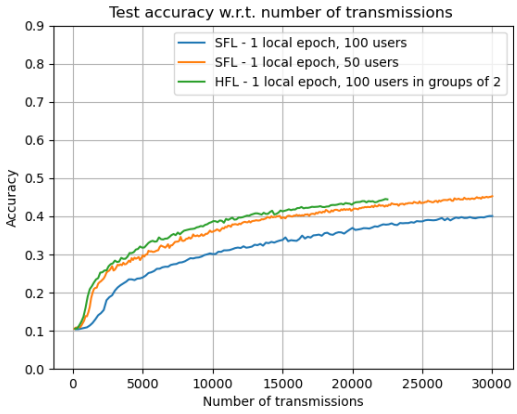
As it is possible to notice, the conclusion drawn for FEMNIST dataset can be repeated also in this case. Indeed, as regards the number of rounds, the hybrid strategy outperforms the two cases of normal FL, which are instead similar, regardless of the number of devices chosen at each global step. On the other hand, performances are comparable when considering time, but the hybrid solution is



(a) Test accuracy w.r.t. rounds



(b) Test accuracy w.r.t. time slots



(c) Test accuracy w.r.t. transmissions

Figure 2.16: CIFAR test accuracy with iid data

still better referring to the required amount of transmissions. However, while the difference is quite marked comparing both methods under the same number of clients chosen each round, the standard solution in which only 50 users are involved in the training is just slightly worse than HFL.

2.6.3 Non-IID data

Also with CIFAR-10, some experiments are conducted on non-iid data. This time, the settings are more extreme with respect to the FEMNIST case, in order to evaluate the goodness of FL itself and the hybrid approach.

The total amount of created users is still 500, having 100 samples each. For every client, data are randomly picked from only one class in such a way that every

class has data spread on an equal number of devices, 50.

Recall that in this case learning rate is set equal to 0.001: indeed, by using a value of 0.01, due to the data setting, the test accuracy curve would have presented more accentuate fluctuations.

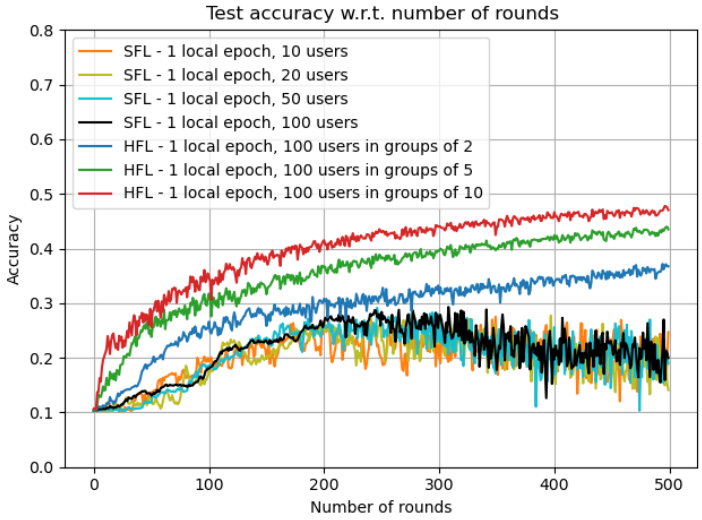
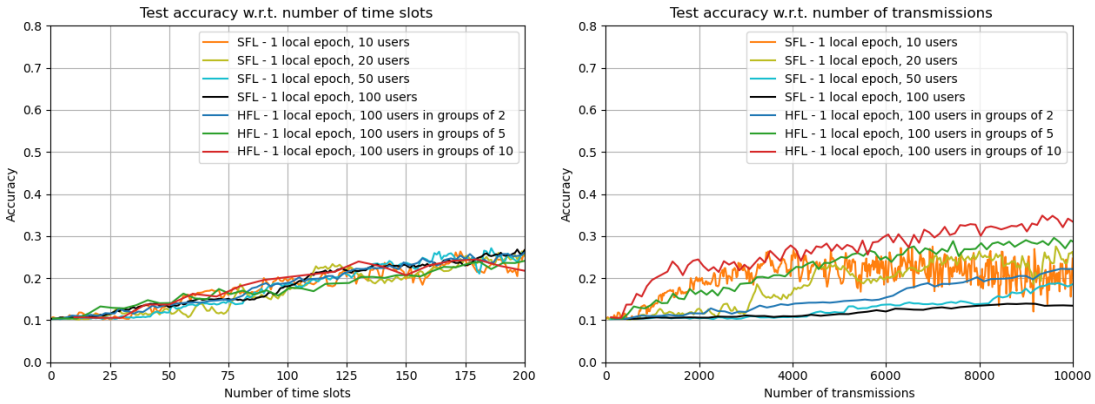


Figure 2.17: CIFAR test accuracy - comparison between SFL and HFL with non-iid data with respect to rounds

Figure 2.17 shows the comparison between SFL and HFL, over 500 rounds: for the first method, the number of users selected at each round is respectively 10, 20, 50 and 100, while for the hybrid strategy the three cases analyzed have all 100 clients per rounds, but grouped in clusters of 2, 5 and 10 members. As it possible to notice, for all the standard method simulations the test accuracy, at a certain point, stop increasing and starts to oscillate, even decreasing a little bit, meaning that with the set parameters the training does not lead to convergence: the reason should be the data distribution, since, as stated, the non-iid configuration is very excessive. On the contrary, all three curves of HFL seem to not suffer this issue, as shown by the growing accuracy; moreover, the performances improve increasing the number of clients per ensemble.

However, for the same reason, also HFL curves may stop improving in future rounds and present divergence issue, which would be though delayed, thus allowing to reach higher accuracy. Therefore, in this case, under the same settings, the proposed method is more stable and can be helpful to achieve better performances when dealing with users having much diverse data.

Besides the convergence problem, the same simulations presented in figure 2.17 are compared with respect to time slots and to the number of transmissions, as shown respectively in figures 2.18a and 2.18b. Note that here, since the test



(a) Test accuracy w.r.t. time slots **(b)** Test accuracy w.r.t. number of transmissions

Figure 2.18: CIFAR test accuracy - comparison between SFL and HFL with non-iid data with respect to time and transmissions

accuracy should be compared in the same range of values, the x-axis of both spans over a determined interval. This implies that the quantity of rounds is no more fixed as in the previous image: this would lead to curves of different lengths, since a round takes a different amount of time and transmissions depending on the number of users picked at each iteration and on the size of the groups for HFL. Anyway, for the first plot, figure 2.18a, the behaviour is comparable among all the simulations, which do not show evident differences.

The most interesting plot is the one showing the test accuracy referred to the number of transmissions, figure 2.18b. Here, except for the normal case with 100 users, all the other curves as coupled considering the number of devices which train in parallel at the same time. Thus, the normal case with 50, 20 and 10 users are respectively associated with the hybrid simulations involving clusters of 2, 5 and 10 members. For each couple, the hybrid version outperforms the corresponding standard version. Note also that the SFL simulation with only 10 users per round is the first curve that manifests a clear sign of divergence: this makes sense, as at each iteration the learning process involves very few devices, which are not enough to train properly the model.

Nevertheless, it seems that a higher group size for HFL causes larger fluctuations; thus, it is desirable to find a solution that can limit them. Having this goal in mind, other simulations are performed, whose results are shown in figures 2.19a, 2.19c and 2.19e, where only the hybrid approach is considered, gathering users in groups of 5, 10 and 20 members. For each of them, two different cases are taken into account: in the first one, ensembles are built randomly, while in the second instance they are made with a criterion. Indeed, for the groups of 5 and 10 devices, every client has complementary datasets with respect to each other, meaning that

all samples of a user belong to a class different from those of other clients. Note also that in the case of 5 members classes of samples are still picked randomly. Finally, for the case of 20 members within a group, there are exactly 2 users per each class. The logical principle for this choice is to avoid clusters with similar data across internal users.

As it is possible to see from figures 2.19a and 2.19e, it seems that curves relative to non-randomly built groups are a bit smoother than their counterparts. Also in this occurrence, simulations are comparable with respect to time, represented in figure 2.19c, even if it seems that performances get slightly worse when enlarging the group dimension.

However, building ensembles with the presented rationale helps to stabilize the test accuracy, despite, especially in the first learning stages, performances are a little lower. In general, grouping users may be a great advantage for the communication side: if one considers a long and continuous training process that aims to further improve the accuracy, also after reaching a threshold that can be acceptable for the examined use case, the hybrid approach would allow to greatly diminish the total of needed transmissions.

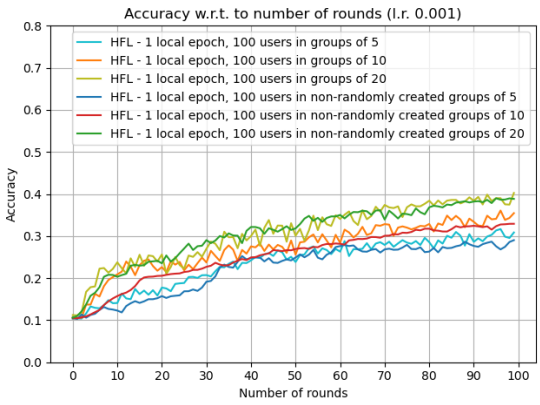
A possible explanation for the smoothing effect with respect to the random selection of the group members should be sought in a previously presented concept. It was already stated that, in non-iid settings, a high number of local epochs for each client can lead to reaching their own local optima, with the risk of worsening the parameters that should optimize the global objective. This explanation can be extended also to groups: if within a single cluster there exist different devices with data belonging to the same class, basically, the group users might have similar cost function and will send to the server an update good just for this kind of objective and not for the general one. Thus, as the members of each ensemble are chosen randomly, it is possible to have many unbalanced groups that can badly affect the global optimization.

Conversely, gathering clients with diverse data leads to having internally balanced groups, meaning that all classes are equally represented. Moreover, clusters are also similar to each other, since all of them are built according to the same logic; therefore, at each round, training data always follow the same distribution: this implies that the new model resulting from the global aggregation is coherent with the previous one, limiting the test accuracy oscillations.

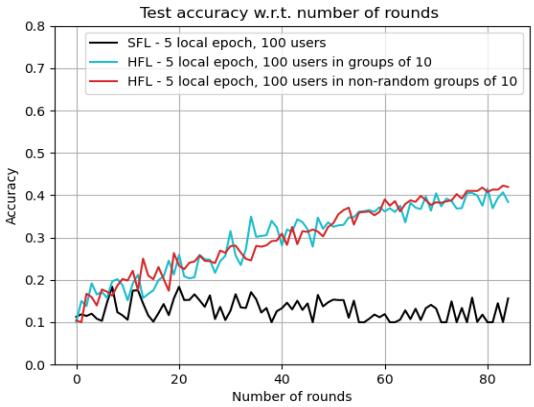
More epochs with non-iid data

As already mentioned, a high number of local epochs can lead the users to optimize their own objective function, negatively effecting the global one. For this reason, a further experiment is run executing 5 local epochs, still with non-iid data.

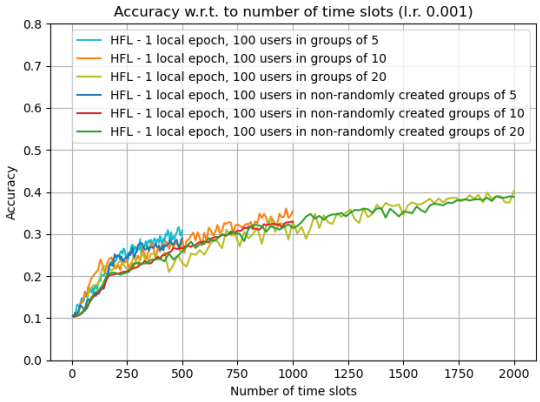
Results as represented in figures 2.19b, 2.19d and 2.19f, which show the curves



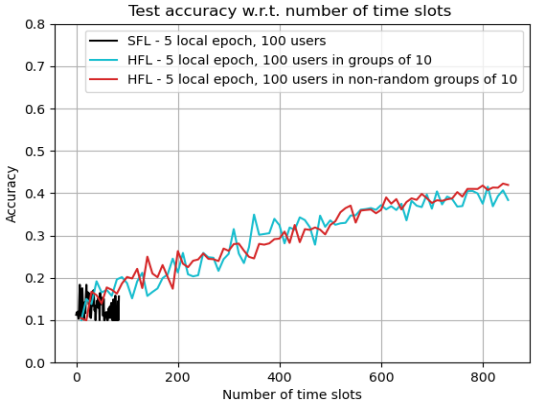
(a) Simulations with 1 epoch w.r.t. rounds



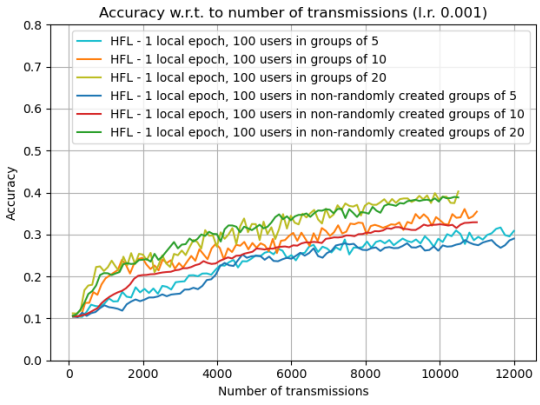
(b) Simulations with 5 epochs w.r.t. rounds



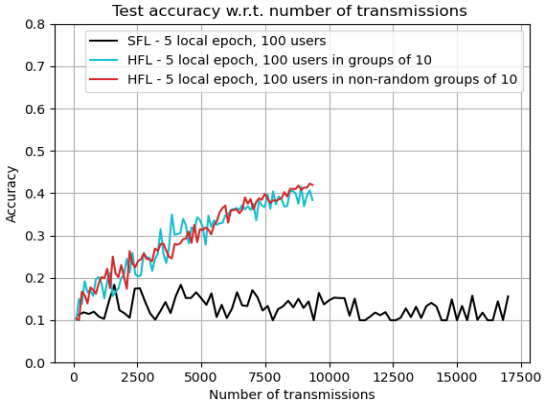
(c) Simulations with 1 epoch w.r.t. time slots



(d) Simulations with 5 epochs w.r.t. time slots



(e) Simulations with 1 epoch w.r.t. number of transmissions



(f) Simulations with 5 epochs w.r.t. number of transmissions

Figure 2.19: CIFAR test accuracy - comparison between SFL and HFL with non-iid data in different settings with respect to rounds, time and number of transmissions

related to SFL and HFL, both with random and non random groups. In all cases the learning rate is set to 0.001 and 100 users are picked each round, with groups of 10 for HFL.

As it is possible to see, the benefits of the hybrid approaches are still the same, both with respect to rounds and to the number of transmissions, while in the considered range of values the normal FL approach does not improve at all. As regards the time slots (figure 2.19d) the curves should be comparable, but it is not evident since the SFL curve is shorter for the reasons explained above.

Also in this case it seems that coordination to build clusters according to the above-mentioned rule leads to some benefits, as the test trend manifests less marked fluctuations than the simulation with randomly created groups. Therefore, from these first experiments, it is possible to state that the orchestration for the formation of the ensembles presents advantages as concerns oscillations.

2.7 Further experiment

So far, all comparisons between SFL and HFL have been made using the same learning parameters. However, a complete validation of the proposed method would require a grid search of the best values for the hyperparameters. Since this operation needs a huge amount of computational resources, for now, it is not executed. To overcome this drawback, it is replicated the same simulation provided in [7], where the authors find the best learning rates related to their simulations settings.

In this work, experiments related to CIFAR-10 dataset are performed dividing samples across 100 clients, in an IID manner. The total of users picked each round is 10, the batch size is set to 50 and the number of local epochs is 5. The used model is the one presented in the TensorFlow tutorial [38], but written using PyTorch framework. The exploited optimizer is the SGD, with learning rate equal to 0.15 and decay, which is applied every round, set to 0.99.

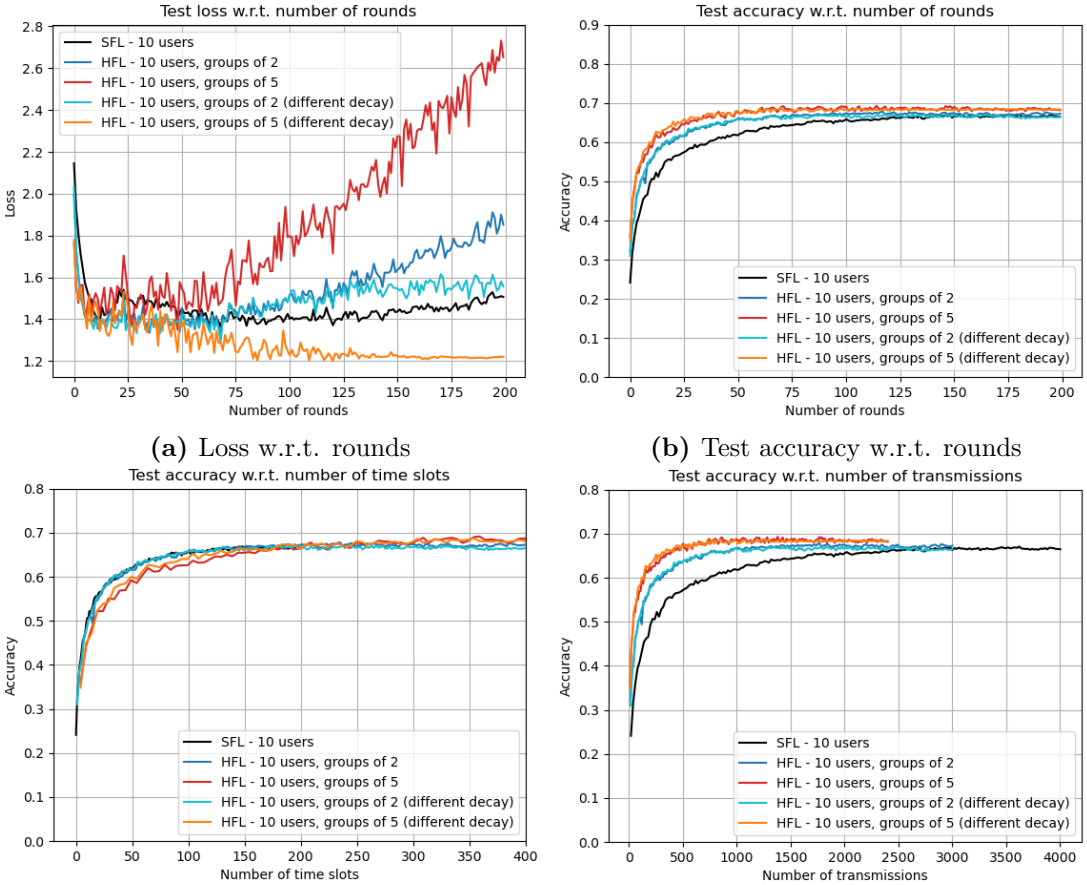
The obtained results are shown in figure 2.20. The SFL final accuracy values are lower with respect to the achieved in [7], but the reason should be due to the data pre-processing, which is not applied in this work, to a smaller number of performed rounds and to the loss. Indeed, it is not clear which kind of function is used, while in this study the *Cross Entropy Loss* is utilized.

Nevertheless, also in this case, the hybrid approach allows achieving higher accuracy. For the latter strategy, two experiments have been executed, with groups size of 2 and 5 respectively. Being the users chosen at each round only 10, these are the only values that are suitable for the simulation.

The case with cluster dimension of 5 grows faster as concerns the rounds and the quantity of transmission and allows reaching higher accuracy after 200 iterations.

As regards the time slots, instead, it is a bit worse, even if at a certain point the curve overtakes the other two.

The experiment involving ensembles of dimension of 2 members is basically the same as SFL for the time slots; the gain is present when considering the number of rounds and of transmissions. This is valid especially for the first steps, while toward the end the accuracy is comparable to the one of SFL.



(a) Loss w.r.t. rounds

(b) Test accuracy w.r.t. rounds

(c) Test accuracy w.r.t. time slots

(d) Test accuracy w.r.t. transmissions

Figure 2.20: Comparison between SFL and HFL with setting of [7]

In this case, also the test loss is plotted, as shown in figure 2.20a, since it is important for a further analysis and to understand the reason why in all plots of figure 2.20 there are two additional curves related to HFL, the ones identified with the label "different decay". Indeed, using a decay of the learning rate each round leads the loss for the hybrid approaches to increase at a certain point, meaning that overfitting may have occurred. In the occurrence with group size equal to 5 this issue is much more evident.

To overcome this drawback, another strategy for the learning rate decay is implemented for HFL, where the decay is no more applied each round, but every time slot. This means that clients training in parallel use all the same value, which is lower with respect to the one applied by the previous users.

As it can be observed, the results applying this method are basically the same for the test accuracy with respect to their counterparts, but are much better for the loss. In fact, the curve corresponding to group size of 5 is always decreasing and becomes steady after 100 rounds, besides being lower than the SFL loss; on the other hand, when clusters include only 2 clients each, this advantage is less noticeable, as the loss increases a bit and finally stabilizes. Nevertheless, in the end its loss is a little higher than the SFL one, which, starting from the 150^{th} iteration, also grows up a little.

To conclude, the hybrid method is beneficial also under the presented settings; however, to prevent the divergence of the loss, it is applied a different learning rate decay strategy, which does not bring any change to the accuracy, but it is useful to stabilize the loss. Thus, this also highlights the importance of the decay value, which is another parameter to be carefully chosen.

Chapter 3

Trajectory Prediction

3.1 Introduction

According to the World Health Organization, every year, around 1.35 million people die due to road traffic crashes, while between 20 and 50 million more people suffer non-fatal injuries, which in many cases lead to disability [39]. This is one of the main reasons for which in the last decades, research has put much effort into studies about Advanced Driver Assistant, ADAS, and autonomous vehicles, which can improve driving safety. Risk assessment is one of the most important topics related to this domain, as it is necessary to handle critical situations.

In [40], the authors use the congestion level and risk thresholds to compute the corresponding risk for the vehicles, in order to safely navigate in the environment. [41] and [42] study risk assessment in occluded environment to increase safety of a given trajectory. While the last two works face the problem using data from real world and modeling the distribution of risk, given the road topology, the observed vehicles and unobserved regions, [43] uses Deep Reinforcement Learning to approach the problem of occluded intersections. The issue with Deep RL is that generally only data coming from simulation are utilized and just a few existing works are then tested in real-world scenarios [44].

However, a lot of works are not related to the direct computation of risk, but concern the trajectory prediction, which can be used for risk assessment, in order to anticipate hazards and avoid them.

[45] is a survey about vehicle behaviour prediction for autonomous vehicles. As this publication dates back to 2014, only traditional approaches are analyzed, such as physics models and simple machine learning techniques. The authors divide methods into three models: physics-based, manoeuvre-based, and interaction-aware. The first approach faces the problem only considering the laws of physics; the second one is based on intended manoeuvre prediction, while the latter takes into

account also interactions among vehicles.

Nevertheless, in the last years, deep learning methods have become popular due to their promising performance in more complex environments compared to the conventional approaches. The latter include many methods, such as the Kalman filter, Monte Carlo Simulation or Hidden Markov Models: in [46], authors state that, while these techniques are accurate for short time predictions, their performances decay for long term forecasting, "which directly affects the accuracy of decision making and path planning". As claimed in [47], the reason is that "these methods typically focus on analyzing objects based on their previous movements and can only be used in simple traffic scenarios with few interactions among vehicles but such methods may not work well in scenarios involving heterogeneous types of vehicles and pedestrians". Evidence of this is also shown in [48] and [49], in which authors empirically prove that Kalman filter is outperformed by all other approaches that are taken into account in the respective works.

3.2 Deep Learning: methods summary

This section is a brief summary of work [50], in which *Mozaffari et al.* present a detailed survey of most recent approaches, providing an in-depth classification based on input representation, output type and used model, with an explanation of the advantages and disadvantages of each method. As regards the input, authors divide methods as belonging to 4 classes:

- history of the target vehicle, TV, that is the agent whose trajectory must be predicted. This approach does not take into account the interactions with the environment and with other agents;
- history of the TV and surrounding vehicles, SVs. This approach uses other vehicles data to consider their influence on the TV, but the interaction with the environment is still missing;
- Bird's Eye View, BEV, that is a map image representing the location of objects and the road topology. This allows to consider the impact of other agents and environment;
- raw sensor data, which leads to higher usage of computational resources.

The first three methods are built on a perception model: given raw data, it must detect and recognize other agents, inferring their position and other useful information. Because of this, it is clear that the performances of the prediction model strongly depend on the accuracy of the first pipeline layer. Relating to the classification made in [45], the last three methods belong to the interaction-aware models.

Then, the authors classify the different approaches also considering the output type, which is divided into four classes:

- Manoeuvre intention, which is the simplest method. However, this technique needs to define a set of possible manoeuvres, factor that limits the power of the approach as the real trajectory may be not included among the predefined ones;

- Unimodal trajectory, as it estimates only one trajectory. Also in this case a set of manoeuvres can be taken into account: even if this presents the above-mentioned drawback, it prevents the result from being the average of two modes, that may be an invalid movement. The latter problem is instead likely when potential intentions are not considered for the prediction;

- Multimodal trajectory, that predicts "one trajectory per behaviour modes (a.k.a. policy/manoeuvre/intention) alongside the mode probability. [50]"

- Occupancy Map, that divides the road map into cells, predicting their occupancy. A clear disadvantage is that accuracy depends on the size of cells.

Finally, as regards the used models, Deep Learning models are the state-of-the-art approaches. The most used prediction methods are the following:

- Recurrent Neural Network, RNN, which are able to capture temporal patterns, but they lack in modelling spatial dependencies;

- Convolutional Neural Network, CNN, that, as opposed to RNN, are good in capturing spatial features, but not temporal ones;

- hybrid methods, that rely on a combination of RNN and CNN, in order to take advantage of both their capabilities;

- Graph Neural network, that models with a graph the structure of agents in traffic.

3.3 Datasets

In this section, it is presented a list of the most popular datasets used for trajectory prediction, divided into two categories. In the first one, data are collected from an external infrastructure, while in the second one they are gathered by the vehicles themselves.

3.3.1 External camera system

The disadvantage of these datasets is that recordings are collected using an external camera system, implying that this type of data is not suitable for approaches that bring the intelligence on smart vehicles, which are now intensely studied. The most used datasets are:

- Interstate 80 Freeway Dataset, I-80, [51], which contains trajectory for vehicles in a US highway, the Interstate 80, for a total of 45 minutes.
- US Highway 101 Dataset [52], US-101: it has the same features as the I-80 dataset, but data are collected in different highway, the Highway 101; together with I-80, it has been developed by the U.S. Department of Transportation for the Next Generation Simulation (NGSIM) program;
- HighD[53]: it provides vehicle trajectories recorded at six different locations on German highways, for a total duration of about 17 hours. Data are collected by using a drone.

All of them contain data related to different traffic conditions. Recently, the research team of the HighD has added other two datasets, named InD [54] and roundD [55]. They contain trajectories of vehicles, pedestrians and bicyclists collected in Germany, respectively in four different intersections and three roundabouts.

3.3.2 Vehicle camera system

In the last years, technological development has lead to a growing usage of sensors on cars, such as cameras and radars, in order to collect data and process them autonomously, allowing the machine itself to sense the environment and take decisions. For this reason, collections containing data recorded by the vehicles have increased their interest for research purposes. The most popular datasets are:

- ApolloScape Trajectory [56], consisting of camera images, LiDAR scanned point clouds and manually annotated trajectories. Data are collected in Beijing, China, for a total duration of about 2 hours and represented scenarios have different lighting and traffic conditions;
- Argoverse Motion Forecasting dataset [57], which includes 320 hours of trajectory data, together with maps encoding geometric and semantic information. Data are represented as 2D birds-eye-view, BEV, and are collected in Miami and Pittsburgh, in different times and weather conditions.
- NuScenes [58], which includes recordings from 2 cities, Boston and Singapore, for a total of almost 6 hours. It provides data from camera and LiDAR, together with 3D bounding boxes;

- Lyft Prediction Dataset [59], which consists of 1118 hours of traffic scenes, captured by 20 self-driving vehicles in Palo Alto. The drawback is that data are collected along a single route, as the authors assume that first applications of self-driving fleets are likely to be used only in few high-demand routes. Data are in BEV format and are enriched with an HD semantic map and high-resolution aerial image of the area: the semantic map contains information about the road, such as lane segments, and other traffic elements, while the aerial map gives spatial information of the environment.

3.4 NGSIM simulations

To the best of my knowledge, to date, no one of the above datasets has been exploited to perform Federated Learning.

In this work, a simple trajectory forecasting model is trained, in order to validate FL also for a use case related to the vehicular domain. To do this, the NGISM US-101 dataset is used.

As already stated, this typology of data collection should not be appropriate for FL, as samples are collected by external cameras. However, with some assumptions, contrary to the sets of the second class, it possible to process data and make them appropriate for federated settings.

The main objective is to validate FL and the proposed approach also with this set, thus the experimented method is straightforward, as explained in the next sections; indeed, providing a method with performances comparable to the current state of the art is out of the scope of this work.

3.4.1 Related works

As previously mentioned, in the literature there are no studies with focus on Federate Learning applied to the NGSIM US-101 dataset, but it is worth citing related works, which apply centralized Deep Learning approaches.

In [60], authors exploit US-101 dataset features related to the target vehicle, TV, that are its position, velocity and class, and other information about its surrounding vehicles, such as the type, the lateral speed, the distance and the time-to-collision with respect to the target vehicle. As concerns the network, the reference architecture consists of an LSTM block followed by three fully connected layers (the last one is the output layer). The objective is to predict the lateral position and the longitudinal speed of future time steps, given as input data windows of 100 samples each, corresponding to the 10 previous seconds.

In [46] a similar approach is followed, but using the I-80 NGSIM dataset. Here, the sequence of past states includes only 50 inputs, thus 5 seconds. The important point of this paper concerns the architecture, as the authors employ two LSTMs.

The first one, fed with data related to lateral features, is able to recognize if a vehicle will change the lane or not, while the second network receives in input the longitudinal features and the output of the first LSTM, aiming to predict the future lateral position and the longitudinal acceleration. Note that in both works the longitudinal position is not forecast since its values can be very large.

In [48] a more complicated model is used, still based on LSTM, which predicts "a multi-modal distribution over future motion". Both the NGSIM US-101 and I-80 are used. As concerns the architecture, the authors develop an LSTM encoder, which encodes the past states of the tracked vehicle and of its six surrounding ones. Then, a maneuver classifier generates a vector encoding the probability of each maneuver given the past trajectory. Finally, the output of these two blocks are stacked together and fed to an LSTM decoder which generates the multi-modal distribution for the position forecasting.

3.4.2 Pre-processing

The NGSIM US-101 datasets provides a collection of vehicles trajectories. For each agent, associated with an ID, there is a sequence of data over time: for every sample, related to a specific instant, the features used in this work are the local lateral and longitudinal position, the velocity, the acceleration, the lane ID, the space and time headway and, finally, information correlated to the vehicles, which are the width, length and the class. The native set does not give the speed and acceleration values with respect to the two directions, but just the scalar value. Also, it only contains the ID of the following and preceding agents, but data about other surrounding vehicles are not provided. Thus, a further pre-processing would be needed to compute the speed and acceleration along the x and y axis and information about contiguous agents: because of this, for now, these features are not used and also the preceding and following vehicles are not taken into account.

At this point, it is fundamental to make data suitable for FL, meaning that it is necessary to simulate that samples are recorded by vehicles and not by an external system. To do this, it is assumed that each agent is able to sense just vehicles in a range of 50 meters: with this supposition, every user is associated with a file containing only data related to those close vehicles; moreover, all values within the individual files are scaled in the range [-5,5], except for global time and vehicle IDs.

However, this procedure may lead to a drawback, not present in the starting dataset. Indeed, a sensed vehicle may be less than 50 m far during two non consecutive time intervals, thus it is necessary to split the two sequences in order to not mix discontinuous samples when giving them in input to the LSTM.

For each data of all files, features of the sensing vehicle, referred to as ego-vehicle (EV), are included: specifically, the x and y position, the speed and the acceleration of EV are attached, such as the difference of these values with respect to those of

the corresponding sample. Of course, the added information refers to the same time instant of the data to which they are attached.

To feed inputs to the LSTM, for all sequences, windows of 100 samples, corresponding to 10 seconds, are extracted. They are obtained with a step of 10, therefore two successive windows have 90 examples in common.

3.4.3 Model

The used model is a particular kind of Recurrent Neural Network, named LSTM, as done in many related works, since they perform well with long sequential data.

The architecture consists of one LSTM layer, followed by 3 fully connected layers, among which the latter is the output one. The hidden and output size of the LSTM is set 256; the dimensions of the dense layers are respectively 256, 131 and 2, which is the number of the output features, x and y. The second FC layer has a size of 131 because it is the result of stacking the output of the first dense block, having size 128, with other three values, which are the width, length and the class of vehicles. These features, as they are fixed, are not given in input to the recurrent layer, which instead learns the temporal patterns. The architecture is almost equal to the one presented in [60].

Given a sequence of 100 samples, the network uses the first 90 data to predict the future 10 states, represented by the x and y positions. For the experiments, the loss function used for the training is the Mean Squared Error, MSE, while the test error consists in the Mean Absolute Error, MAE, in meters, along the lateral and longitudinal position. The batch size is equal to 32, while the utilized optimizer is the Adam: the weight decay is set to 0.1, the parameters β_1 and β_2 are respectively 0.9 and 0.999 and the starting learning rate is equal to 10^{-3}, with exponential decay of 0.95 each round.

3.4.4 HFL

As concerns the hybrid approach, a new challenge arises. Indeed, it is not possible anymore to build groups in a random way, but it is important to take into account also the position of vehicles, as it would be impossible to directly transmit the model to a far user. Therefore, it is proposed an alternative to overcome this issue: a vehicle that has to share its model, should send it to the closest vehicle that has not been peaked yet for training during that round.

However, in this case, a client may not have any close agent, thus it is unable to keep propagating the model. For this reason, a solution for this problem is to permit also smaller groups. In this way, a device that does not have another machine to which it can transmit its updated parameters, directly sends them to the server. Nevertheless, both methodologies are implemented in the simulations

for HFL: with groups randomly created and with the distance criterion, allowing smaller ensembles.

3.5 Results

After the pre-processing step, a total of 5986 vehicles are obtained; among them, 90%, corresponding to 5387 users, is used for training and the remaining 10%, equivalent to 599 clients, for the test phase. Note that the final amount of vehicles is lower than the ones actually present in the NGSIM US-101; indeed, vehicles that have no sequences of 100 consecutive samples or no neighbours (for the model transmission in the hybrid approach) are discarded.

During training, at every round, 500 devices are peaked, both for SFL and HFL, in which groups have size of maximum 10 members.

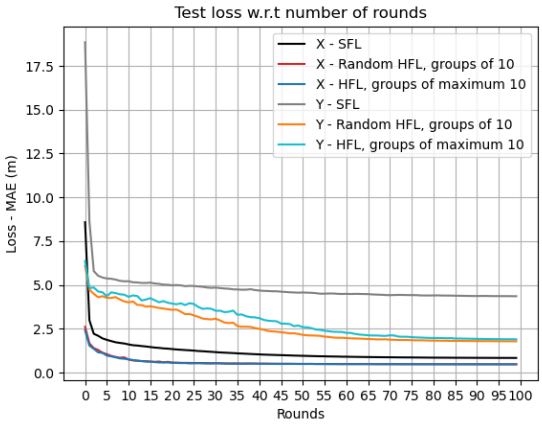
Performing different simulations, features of class, width and length of the target vehicle badly affect the training; thus, these data are dropped, but certainly they should be better analyzed. This implies that the last but one fully connected layer has no more size equal to 131, but of 128 instead.

Also, another change has been made, as the local position x and y of the target vehicle are no more exploited. This decision has been made for two reasons. First, the TV position values are related to a local reference point: this makes sense for the original dataset, since an external infrastructure is used and it is possible to relate all vehicles position with respect to the place in which camera start to collect information. For the federated approach, instead, defining a local reference would not be a good strategy, as it would be difficult to handle when taking into account vehicles in different geographical spots. Second, the local longitudinal position spans over a quite large interval of values, while the difference between TV and EV is, of course, limited, as only agents in a range of 50 meters are sensed by assumption; this also leads to having a lower error. Therefore, predictions are computed referring to the position of the respective ego-vehicles.

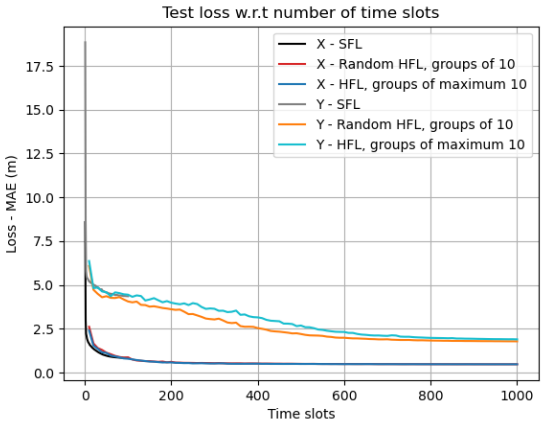
Figures 3.1a, 3.1b and 3.1c show the test MAE along the lateral and longitudinal direction (x and y), for SFL and HFL, both with random and non-random created ensembles. As it is possible to notice, the y error is bigger regardless of the method; this makes sense, as the longitudinal values are generally higher, thus leading to greater error.

Concerning the comparison between SFL and HFL, the latter allows to reach much lower loss values considering the same round or fixing the number of transmissions. Instead, regarding the time, in the range from 0 to 100, the trend is the same; the only difference is that in SFL the starting value is much higher, but after a very quick drop, curves are comparable.

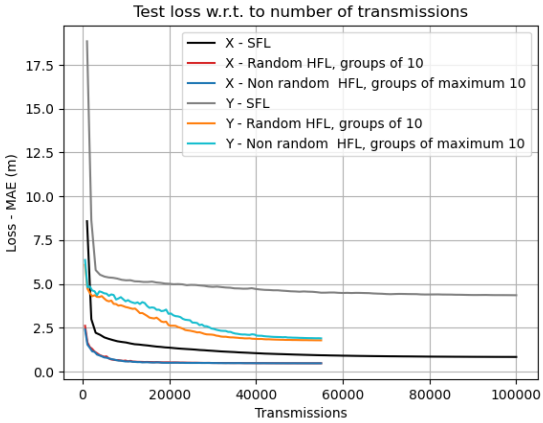
Moreover, both for the x and y errors, the random method is slightly better



(a) Test accuracy w.r.t. rounds



(b) Test accuracy w.r.t. time slots



(c) Test accuracy w.r.t. transmissions

Figure 3.1: NGSIM US-101 test accuracy

(for the lateral position the difference is not noticeable from the plots only due to the used scale). The reason could be that in the random case all ensembles have the same dimension, while this is not true for the other strategy. This means that the models obtained by some ensembles may be trained on fewer data, thus negatively affecting the aggregation performed by the controller, which results in worse performances over the rounds.

However, errors are quite high, even for HFL, which shows MAE around 0.5 m and 2 m, respectively for the lateral and longitudinal positions. To improve the results, it would be necessary to train for more epochs and also consider the surrounding vehicles, as done in other works, because the behavior of each agent is strongly dependent on the other ones.

For sure, a deeper study of the architecture and the features to use is needed: as experimentally found in [60], changing one of these factors leads to very different results. Despite these observations, the benefits of the hybrid algorithm are shown also in this case.

Chapter 4

Conclusions

In this work Federated Learning is presented, an emerging method of Machine Learning field. The reasons for the increasing interest in this technique are mainly two. Indeed, it allows to train a model keeping data within the devices, preventing from sharing them with a central server: first, this implies that data are no more transmitted over the network, process which, due to their huge quantity, could lead to a communication bottleneck, and, second, it increases users privacy.

Then, a novel strategy, named Hybrid Federated Learning, is introduced: it brings a modification to the vanilla algorithm of FL.

Both methods are tested with three datasets, FEMNIST, CIFAR and the NGSIM US-101, under different conditions: specifically, simulations are performed with diverse number of local epochs and taking into account both iid and non-iid data among users samples.

The obtained results are promising and show that the proposed strategy works and, with respect to the standard approach, considering an equal number of users picked at every iteration, it leads to the following advantages:

- the hybrid approach outperforms the SFL method in terms of round;
- as concerns the communication side, it requires fewer transmissions to reach the same accuracy;
- still regarding communication, there is also a major benefit, as it allows to rely on device-to-device communication, thus limiting the bottleneck due to the transmissions with the central server.

With respect to time, instead, test accuracy trends of the two strategies are comparable.

The improvements are also valid in the case of extreme non-iid data, in which HFL shows better robustness, especially with a high number of local epochs. Moreover, coordination to build groups seems to be useful in terms of oscillations.

4.1 Future work

Results are obtained considering the same training parameters, thus the first step to fully validate the novel approach is to perform an extensive research of the best values for both of them and make the comparison. This process, though, would be very time consuming, since it would be necessary to try with different learning rates, batch size and, for HFL, also the group dimension. For the latter, as it is possible to apply the learning rate decay while training the model, maybe it would also be possible to change dynamically the size of the ensembles during the learning.

Then, to further validate the proposed method, it may be interesting to use the state-of-the art models for the given datasets and apply it in more complicated scenarios, which require developing finer neural network architectures. This would be useful to also estimate the bandwidth gain when using the hybrid strategy, as this computation is not performed due to the used models, which are small, thus not used in practice.

Moreover, it would be worth deepening on the time aspect, taking into account the time needed for the weights transmission; this can be done either theoretically or, even better, by implementing on a real platform both the standard FL and the hybrid approach, in order to compare them.

Related to the transport domain, it would be interesting to address the problem of mobility, focusing on its impact on the data collection and, for HFL, on the model transmission among vehicles. Specifically for the hybrid strategy, it would also be worth to better analyze the effects of the strategy to build groups, to understand if and which can be the advantages of coordination.

On the other side, related to the trajectory prediction model, a possible extension concerns the use of information related to the surrounding agents of the target vehicle; finally, it is possible to use the *HighD* dataset [53], which provides more precise data and also information about close vehicles of the TV, without needing further pre-processing.

Bibliography

[1] Adam Grzywaczewski. *Training AI for Self-Driving Vehicles: the Challenge of Scale*. URL: https://developer.nvidia.com/blog/training-self-driving-vehicles-challenge-scale (cit. on p. 1).

[2] Chris Mellor. *Data storage estimates for intelligent vehicles vary widely*. URL: https://blocksandfiles.com/2020/01/17/connected-car-data-storage-estimates-vary-widely (cit. on p. 1).

[3] *Autonomous Vehicles: The Race is On*. https://www.accenture.com/_acnmedia/pdf-73/accenture-autonomous-vehicles-the-race-is-on.pdf (cit. on p. 1).

[4] Wei Yang Bryan Lim, Nguyen Cong Luong, Dinh Thai Hoang, Yutao Jiao, Ying-Chang Liang, Qiang Yang, Dusit Niyato, and Chunyan Miao. *Federated Learning in Mobile Edge Networks: A Comprehensive Survey*. 2020. arXiv: 1909.11875 [cs.NI] (cit. on pp. 1, 3, 6, 7, 11).

[5] Qiang Yang, Yang Liu, Tianjian Chen, and Yongxin Tong. *Federated Machine Learning: Concept and Applications*. 2019. arXiv: 1902.04885 [cs.AI] (cit. on pp. 2, 6, 8).

[6] Shiqiang Wang, Tiffany Tuor, Theodoros Salonidis, Kin K. Leung, Christian Makaya, Ting He, and Kevin Chan. *Adaptive Federated Learning in Resource Constrained Edge Computing Systems*. 2019. arXiv: 1804.05271 [cs.DC] (cit. on p. 2).

[7] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. *Communication-Efficient Learning of Deep Networks from Decentralized Data*. 2017. arXiv: 1602.05629 [cs.LG] (cit. on pp. 2, 4, 5, 7, 11, 12, 34, 35).

[8] Tian Li, Anit Kumar Sahu, Ameet Talwalkar, and Virginia Smith. «Federated Learning: Challenges, Methods, and Future Directions». In: *IEEE Signal Processing Magazine* 37.3 (2020), pp. 50–60. ISSN: 1053-5888, 1558-0792. DOI: 10.1109/MSP.2020.2975749. URL: https://ieeexplore.ieee.org/document/9084352/ (cit. on p. 2).

[9] D. Ye, R. Yu, M. Pan, and Z. Han. «Federated Learning in Vehicular Edge Computing: A Selective Model Aggregation Approach». In: *IEEE Access* 8 (2020), pp. 23920–23935 (cit. on p. 3).

[10] Yuris Mulya Saputra, Dinh Thai Hoang, Diep N. Nguyen, and Eryk Dutkiewicz. *Dynamic Federated Learning-Based Economic Framework for Internet-of-Vehicles*. 2021. arXiv: 2101.00191 [cs.NI] (cit. on p. 3).

[11] Ahmet M. Elbir, Burak Soner, and Sinem Coleri. *Federated Learning in Vehicular Networks*. 2020. arXiv: 2006.01412 [eess.SP] (cit. on p. 4).

[12] Seyhan Ucar, Takamasa Higuchi, Chang-Heng Wang, Duncan Deveaux, Jérôme Härri, and Onur Altintas. «Vehicular Knowledge Networking and Application to Risk Reasoning». In: *Proceedings of the Twenty-First International Symposium on Theory, Algorithmic Foundations, and Protocol Design for Mobile Networks and Mobile Computing*. Mobihoc '20. New York, NY, USA: Association for Computing Machinery, 2020, pp. 351–356. ISBN: 9781450380157. DOI: 10.1145/3397166.3413467 (cit. on p. 4).

[13] Duncan Deveaux, Takamasa Higuchi, Seyhan Uçar, Jérôme Härri, and Onur Altintas. *A Definition and Framework for Vehicular Knowledge Networking*. 2020. arXiv: 2005.14505 [cs.NI] (cit. on p. 4).

[14] Tian Li, Anit Kumar Sahu, Manzil Zaheer, Maziar Sanjabi, Ameet Talwalkar, and Virginia Smith. *Federated Optimization in Heterogeneous Networks*. 2020. arXiv: 1812.06127 [cs.LG] (cit. on pp. 5, 11).

[15] Yue Zhao, Meng Li, Liangzhen Lai, Naveen Suda, Damon Civin, and Vikas Chandra. *Federated Learning with Non-IID Data*. 2018. arXiv: 1806.00582 [cs.LG] (cit. on p. 6).

[16] Jakub Konečný, H. Brendan McMahan, Felix X. Yu, Peter Richtárik, Ananda Theertha Suresh, and Dave Bacon. *Federated Learning: Strategies for Improving Communication Efficiency*. 2017. arXiv: 1610.05492 [cs.LG] (cit. on pp. 6, 11).

[17] Martín Abadi, Andy Chu, Ian Goodfellow, H. Brendan McMahan, Ilya Mironov, Kunal Talwar, and Li Zhang. «Deep Learning with Differential Privacy». In: *Proceedings of the 2016 ACM SIGSAC Conference on Computer and Communications Security* (2016). arXiv: 1607.00133, pp. 308–318. DOI: 10.1145/2976749.2978318. URL: http://arxiv.org/abs/1607.00133 (cit. on p. 6).

[18] Wikipedia contributors. *Secure multi-party computation — Wikipedia, The Free Encyclopedia*. 2021. URL: https://en.wikipedia.org/wiki/Secure_multi-party_computation (cit. on p. 6).

[19] *TensorFlow Federated: Machine Learning on Decentralized Data*. https://www.tensorflow.org/federated (cit. on p. 7).

[20] Theo Ryffel, Andrew Trask, Morten Dahl, Bobby Wagner, Jason Mancuso, Daniel Rueckert, and Jonathan Passerat-Palmbach. *A generic framework for privacy preserving deep learning*. 2018. arXiv: 1811.04017 [cs.LG] (cit. on p. 7).

[21] Sebastian Caldas, Sai Meher Karthik Duddu, Peter Wu, Tian Li, Jakub Konečný, H. Brendan McMahan, Virginia Smith, and Ameet Talwalkar. *LEAF: A Benchmark for Federated Settings*. 2019. arXiv: 1812.01097 [cs.LG] (cit. on p. 7).

[22] Chaoyang He et al. *FedML: A Research Library and Benchmark for Federated Machine Learning*. 2020. arXiv: 2007.13518 [cs.LG] (cit. on p. 7).

[23] Andrew Hard, Kanishka Rao, Rajiv Mathews, Swaroop Ramaswamy, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. *Federated Learning for Mobile Keyboard Prediction*. 2019. arXiv: 1811.03604 [cs.CL] (cit. on p. 7).

[24] Wikipedia contributors. *Homomorphic encryption — Wikipedia, The Free Encyclopedia*. 2021. URL: https://en.wikipedia.org/wiki/Homomorphic_encryption (cit. on p. 7).

[25] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. *EMNIST: an extension of MNIST to handwritten letters*. 2017. arXiv: 1702.05373 [cs.CV] (cit. on p. 7).

[26] Alec Go, Richa Bhayani, and Lei Huang. «Twitter sentiment classification using distant supervision». In: *Processing* 150 (Jan. 2009) (cit. on p. 7).

[27] *The Complete Works of William Shakespeare*. https://www.gutenberg.org/ebooks/100 (cit. on p. 7).

[28] Ziwei Liu, Ping Luo, Xiaogang Wang, and Xiaoou Tang. «Deep Learning Face Attributes in the Wild». In: *Proceedings of International Conference on Computer Vision (ICCV)*. Dec. 2015 (cit. on p. 7).

[29] Praneeth Vepakomma, Otkrist Gupta, Tristan Swedish, and Ramesh Raskar. *Split learning for health: Distributed deep learning without sharing raw patient data*. 2018. arXiv: 1812.00564 [cs.LG] (cit. on p. 8).

[30] Otkrist Gupta and Ramesh Raskar. *Distributed learning of deep neural network over multiple agents*. 2018. arXiv: 1810.06060 [cs.LG] (cit. on p. 8).

[31] István Hegedűs, Gábor Danner, and Márk Jelasity. «Gossip Learning as a Decentralized Alternative to Federated Learning». In: *Distributed Applications and Interoperable Systems*. Ed. by José Pereira and Laura Ricci. Vol. 11534. Cham: Springer International Publishing, 2019, pp. 74–90. ISBN: 978-3-030-22495-0 978-3-030-22496-7. DOI: 10.1007/978-3-030-22496-7_5. URL: http://link.springer.com/10.1007/978-3-030-22496-7_5 (visited on 10/03/2020) (cit. on p. 8).

[32] L. Giaretta and Š. Girdzijauskas. «Gossip Learning: Off the Beaten Path». In: *2019 IEEE International Conference on Big Data (Big Data)*. 2019, pp. 1117–1124. DOI: 10.1109/BigData47090.2019.9006216 (cit. on p. 8).

[33] Jinliang Yuan, Mengwei Xu, Xiao Ma, Ao Zhou, Xuanzhe Liu, and Shangguang Wang. *Hierarchical Federated Learning through LAN-WAN Orchestration*. 2020. arXiv: 2010.11612 [cs.LG] (cit. on pp. 9, 26).

[34] Lumin Liu, Jun Zhang, S. H. Song, and Khaled B. Letaief. *Client-Edge-Cloud Hierarchical Federated Learning*. 2019. arXiv: 1905.06641 [cs.NI] (cit. on p. 9).

[35] Mehdi Salehi Heydar Abad, Emre Ozfatura, Deniz Gunduz, and Ozgur Ercetin. *Hierarchical Federated Learning Across Heterogeneous Cellular Networks*. 2019. arXiv: 1909.02362 [cs.LG] (cit. on p. 9).

[36] Shaoxiong Ji. «A PyTorch Implementation of Federated Learning». In: (Mar. 2018). DOI: 10.5281/zenodo.4321561 (cit. on p. 15).

[37] Alex Krizhevsky. «Learning Multiple Layers of Features from Tiny Images». In: *University of Toronto* (May 2012) (cit. on p. 26).

[38] *Convolutional Neural Network (CNN)*. https://www.tensorflow.org/tutorials/images/cnn (cit. on p. 34).

[39] World Health Organization. *Road traffic injuries*. 2020. URL: https://www.who.int/news-room/fact-sheets/detail/road-traffic-injuries (cit. on p. 37).

[40] Alyssa Pierson, Wilko Schwarting, Sertac Karaman, and Daniela Rus. «Learning Risk Level Set Parameters from Data Sets for Safer Driving». In: *2019 IEEE Intelligent Vehicles Symposium (IV)*. 2019, pp. 273–280. URL: https://ieeexplore.ieee.org/document/8813842/ (cit. on p. 37).

[41] Ming-Yuan Yu, Ram Vasudevan, and Matthew Johnson-Roberson. «Occlusion-Aware Risk Assessment for Autonomous Driving in Urban Environments». In: *IEEE Robotics and Automation Letters* 4 (2019). arXiv: 1809.04629, pp. 2235–2241. URL: http://arxiv.org/abs/1809.04629 (cit. on p. 37).

[42] Piotr Franciszek Orzechowski, Annika Meyer, and Martin Lauer. «Tackling Occlusions & Limited Sensor Range with Set-based Safety Verification». In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)* (2018). arXiv: 1807.01262, pp. 1729–1736. URL: http://arxiv.org/abs/1807.01262 (cit. on p. 37).

[43] David Isele, Reza Rahimi, Akansel Cosgun, Kaushik Subramanian, and Kikuo Fujimura. «Navigating occluded intersections with autonomous vehicles using deep reinforcement learning». In: *2018 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE. 2018, pp. 2034–2039 (cit. on p. 37).

[44] Ammar Haydari and Yasin Yilmaz. *Deep Reinforcement Learning for Intelligent Transportation Systems: A Survey*. 2020. arXiv: 2005.00935 [cs.LG] (cit. on p. 37).

[45] Stéphanie Lefèvre, Dizan Vasquez, and Christian Laugier. «A survey on motion prediction and risk assessment for intelligent vehicles». In: *ROBOMECH Journal* 1 (2014), p. 1. URL: http://www.robomechjournal.com/content/1/1/1 (cit. on pp. 37, 38).

[46] Long Xin, Pin Wang, Ching-Yao Chan, Jianyu Chen, Shengbo Eben Li, and Bo Cheng. *Intention-aware Long Horizon Trajectory Prediction of Surrounding Vehicles using Dual LSTM Networks*. 2019. arXiv: 1906.02815 [cs.LG] (cit. on pp. 38, 41).

[47] Xin Li, Xiaowen Ying, and Mooi Choo Chuah. *GRIP++: Enhanced Graph-based Interaction-aware Trajectory Prediction for Autonomous Driving*. 2020. arXiv: 1907.07792 [cs.CV] (cit. on p. 38).

[48] Nachiket Deo and Mohan M. Trivedi. «Multi-Modal Trajectory Prediction of Surrounding Vehicles with Maneuver based LSTMs». In: *2018 IEEE Intelligent Vehicles Symposium (IV)* (June 2018). DOI: 10.1109/ivs.2018.8500493. URL: http://dx.doi.org/10.1109/IVS.2018.8500493 (cit. on pp. 38, 42).

[49] Henggang Cui, Vladan Radosavljevic, Fang-Chieh Chou, Tsung-Han Lin, Thi Nguyen, Tzu-Kuo Huang, Jeff Schneider, and Nemanja Djuric. *Multimodal Trajectory Predictions for Autonomous Driving using Deep Convolutional Networks*. 2019. arXiv: 1809.10732 [cs.RO] (cit. on p. 38).

[50] Sajjad Mozaffari, Omar Y. Al-Jarrah, Mehrdad Dianati, Paul Jennings, and Alexandros Mouzakitis. «Deep Learning-Based Vehicle Behavior Prediction for Autonomous Driving Applications: A Review». In: *IEEE Transactions on Intelligent Transportation Systems* (2020), pp. 1–15. ISSN: 1558-0016. DOI: 10.1109/tits.2020.3012034. URL: http://dx.doi.org/10.1109/TITS.2020.3012034 (cit. on pp. 38, 39).

[51] U.S. Federal Highway Administration. *U.S. Interstate 80 Freeway Dataset*.

2006. URL: https://www.fhwa.dot.gov/publications/research/operations/06137/ (cit. on p. 40).

[52] U.S. Federal Highway Administration. *U.S. Highway 101 Dataset*. 2007. URL: https://www.fhwa.dot.gov/publications/research/operations/07030/index.cfm (cit. on p. 40).

[53] Robert Krajewski, Julian Bock, Laurent Kloeker, and Lutz Eckstein. «The highD Dataset: A Drone Dataset of Naturalistic Vehicle Trajectories on German Highways for Validation of Highly Automated Driving Systems». In: *2018 21st International Conference on Intelligent Transportation Systems (ITSC)*. 2018, pp. 2118–2125. DOI: 10.1109/ITSC.2018.8569552 (cit. on pp. 40, 48).

[54] Julian Bock, Robert Krajewski, Tobias Moers, Steffen Runde, Lennart Vater, and Lutz Eckstein. «The inD Dataset: A Drone Dataset of Naturalistic Road User Trajectories at German Intersections». In: 2019 (cit. on p. 40).

[55] Robert Krajewski, Tobias Moers, Julian Bock, Lennart Vater, and Lutz Eckstein. «The rounD Dataset: A Drone Dataset of Road User Trajectories at Roundabouts in Germany». submitted (cit. on p. 40).

[56] Yuexin Ma, Xinge Zhu, Sibo Zhang, Ruigang Yang, Wenping Wang, and Dinesh Manocha. «Trafficpredict: Trajectory prediction for heterogeneous traffic-agents». In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 33. 01. 2019, pp. 6120–6127 (cit. on p. 40).

[57] Ming-Fang Chang et al. «Argoverse: 3d tracking and forecasting with rich maps». In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2019, pp. 8748–8757 (cit. on p. 40).

[58] Holger Caesar et al. *nuScenes: A multimodal dataset for autonomous driving*. 2020. arXiv: 1903.11027 [cs.LG] (cit. on p. 40).

[59] John Houston, Guido Zuidhof, Luca Bergamini, Yawei Ye, Long Chen, Ashesh Jain, Sammy Omari, Vladimir Iglovikov, and Peter Ondruska. *One Thousand and One Hours: Self-driving Motion Prediction Dataset*. 2020. arXiv: 2006.14480 [cs.CV] (cit. on p. 41).

[60] Florent Altché and Arnaud de La Fortelle. *An LSTM Network for Highway Trajectory Prediction*. 2018. arXiv: 1801.07962 [cs.RO] (cit. on pp. 41, 43, 46).