Federated Learning Approaches for Human Mobility Models

Master's Thesis submitted to the
Faculty of Informatics of the *Università della Svizzera Italiana*
in partial fulfillment of the requirements for the degree of
Master of Science in Informatics

presented by
Alex Caselli

under the supervision of
Prof. Marc Langheinrich
co-supervised by
Prof. Silvia Santini

October 2020

I certify that except where due acknowledgement has been given, the work presented in this thesis is that of the author alone; the work has not been submitted previously, in whole or in part, to qualify for any other academic award; and the content of the thesis is the result of work which has been carried out since the official commencement date of the approved research program.

Alex Caselli
Lugano, 15 October 2020

To my beloved family, who has constantly supported me. To my friends and fellow adventures, who have shared with me this long journey. To my supervisors, for the essential help provided for the development of this thesis. Thank you.

Abstract

Thanks to the proliferation of smart devices, such as smartphones and wearables, which are equipped with computation, communication and *sensing* capabilities, a plethora of new location-based services and applications are available for the users at any time and everywhere. Understanding human mobility has gain importance to offer better services able to provide valuable products to the user whenever it's required. The ability to predict when and where individuals will go next allows enabling smart recommendation systems or a better organization of resources such as public transport vehicles or taxis. Network providers can predict future activities of individuals and groups to optimize network handovers, while transport systems can provide more vehicles or lines where required, reducing waiting time and discomfort to their clients.

The representation of the movements of individuals or groups of mobile entities are called *human mobility models*. Such models replicate real human mobility characteristics, enabling to simulate movements of different individuals and infer their future whereabouts. The development of these models requires to collect in a centralized location, as a server, the information related to the users' locations. Such data represents sensitive information, and the collection of those threatens the privacy of the users involved. The recent introduction of *federated learning*, a privacy-preserving approach to build machine and deep learning models, represents a promising technique to solve the privacy issue. Federated learning allows mobile devices to contribute with their private data to the model creation without sharing them with a centralized server. In this thesis, we investigate the application of the federated learning paradigm to the field of human mobility modelling.

Using three different mobility datasets, we first designed and developed a robust human mobility model by investigating different classes of neural networks and the influence of demographic data over models' performance. Second, we applied federated learning to create a human mobility model based on deep learning which does not require the collection of users' mobility traces, achieving promising results on two different datasets. Users' data remains so distributed over the big number of devices which have generated them, while the model is shared and trained among the server and the devices.

Furthermore, the developed federated model has been the subject of different analyses including: the effects of sparse availability of the clients; The communication costs required by federated settings; The application of transfer-learning techniques and model refinement through federated learning and, lastly, the influence of differential privacy on the model's prediction performance, also called *utility*.

Contents

Figures

Tables

Chapter 1

Introduction

Mobile and wearable devices, such as smartphones and smartwatches, are everywhere and more popular than ever with almost 368 millions of smartphones sold just in 2019[12]. Furthermore, the recent development of new mobile technologies, such as the new standard for cellular networks 5G, enables such devices to communicate and exchange data even faster than common cabled connections.

Mobile devices are, by their nature, able to move or to be moved, usually alongside their users. This feature, together with the capability of collecting various types of data using the sensors usually embedded in the devices, allows users and services installed on the devices to share and use a huge amount of information to enable additional features. For instance, GPS and Wi-Fi can be used to locate the user and automate actions, like activating the home heating system whenever the user is thought to come back home[13][14], or suggest when to leave the house to get to work in time by also considering traffic and public transport timetables[15].

Devices' location information represents one nature of this precious data, where the location of a mobile device very often represents the location of its user. Information related to the location of the device can be acquired in different ways and using different sensors including Wi-Fi, GPS, Bluetooth and cellular antennas. Further, it is also common for users to geo-tag themselves selecting the desired location using one of the most common social platforms and applications (e.g. Instagram, Twitter or Foursquare).

Location data has many usages. Tracking and navigation systems use the user's location to provide suggestions for routing, while services and network providers can use the collected mobility traces to predict future activities of individuals and groups to optimize network handovers, automate tasks, or to simulate realistic network loads. Furthermore, users' location data can be used by companies such as Google to perform targeted advertisement of products or services[16].

In the last year, due to the global pandemic caused by the new coronavirus SARS-CoV-2, important services such as public transports have been affected by the limitations required to protect the customers, creating difficulties for both the managers and the clients themselves. The understanding of human mobility has become more and more important to better manage such kind of services, especially in urban environments where the virus spreads easier.

The ability to predict which places people will visit and when becomes crucial to provide efficient services which can reduce the time spent by people close to each other and prevent crowds by boosting services in the rush hours of the day. Even more important is to apply

contact tracing, by finding people who have been close with a tested positive person in the previous days in order to perform accurate testing of the population.

Several countries introduced contact tracing services and applications. The South Korean government, instead of using an app-based contact tracing system, has opted for a centralized option which involved many different data sources to track its citizens such as the smartphones GPS data, credit card transactions and CCTV footage[17]. This information is used to reconstruct the mobility traces of individuals who tested positive for the new coronavirus to discover other potentially infected persons.

The privacy concerns about this system adopted also from other closest countries, become even bigger considering the app "Corona 100m". The app, developed and used in South Korea and ranked as sixth of the top 15 downloaded apps on the country's Google Play app store[18], allows people to see when people have been tested positive to Covid-19 together with their personal information. The information includes the patient's nationality, gender, age and places that they have recently visited, exposing so private information, which could be misused for unlawful or unauthorized analyses.

The user's location is by its nature very sensitive and it is so protected by privacy laws and regulations in many countries. Both the collection and use of location data is usually strictly limited for both service providers and governments. The location of a user could be exploited with malicious purposes such as allowing criminals to assault a specific individual; Carry out political and religious persecution or to profile the user and infer personal information. Such information could be used to preclude possible benefits such as cheaper health care by making assumptions on its health state or personal habits.[19]

Human mobility is also highly predictable, researches have shown that movements of individuals present a potential 93% mean predictability and are characterized by a low entropy[20]. This characteristic introduces the possibility of using the understanding and the regularities of the human mobility, together with publicly available mobility data, to develop simulation and prediction systems of individuals or populations, often called human mobility models.

In the following, we describe fundamental concepts around human mobility models.

1.1 Human Mobility Models

Hess et al.[21] define a human mobility model as a *"simplified representation of the movement of single or groups of mobile entities in a given context, primarily the spatial environment"*. Such a model aims to reproduce or generate mobility traces by an understanding of the underline patterns that regulate the mobility of humans.

Human mobility models may support many different services and applications. Their ability to simulate movements of individuals, crowds or specific groups of people can be used by governments, services and internet providers. Network infrastructures can be designed properly considering whether more clients will group together and so provide larger bandwidth or more antennas. Public institutions can take better decision when designing urban areas thanks to the knowledge of the movements of the citizens, while public transports and bike-sharing companies can provide more vehicles or lines where more required, reducing waiting time and discomfort to their clients.

Munjal et al.[22] categorized mobility models into two main different groups: synthetic and trace-based models.

While synthetic models are relatively easy to use and analyze due to their mathematical definition, models such as the widely used RWP (Random Waypoint), they are often too simplistic to capture relevant characteristics of real human mobility[23]. Entities in synthetic models move randomly following probability distributions. At each time, a new destination is sampled for a transition from the current location and so entities move independently.

Trace-based models provide instead more realistic outcomes but require having real large-scale traces data. Examples of datasets containing such traces include, e.g., the CRAWDAD archive[24], the Levy-Walk repository[25], the Nokia LDCC dataset, created during the Nokia Lausanne Data Collection Campaign[26]. While these are examples of datasets collected as part of research activities, there exist also large dataset collected by public institutions and municipals such as the NYC taxi datasets[27] released by the NYC Taxi and Limousine Commission (TLC), or the Singapore's Land Transport Authority[28]. The mobility traces are then used to extrapolate the key characteristics and patterns of human mobility and replicate them through human mobility models.

Noulas et al. in 2012 identified a universal law for human mobility which isolates as a key component the rank-distance, factoring in the number of places between origin and destination. While this law well describes the underlined mechanisms of human movements, it also fails to capture the fact that those movements take place in contexts. Both the geographical and temporal contexts can influence human mobility, such as different different seasons and weather conditions may affect the way people move.

The majority of mobility models proposed by the literature are based on offline learning, which requires to have collected all the data needed to build the model so then, once the learning phase is completed, the model building phase finishes and it does not need new data to operate.

The assumption made by offline learning of having all the necessary data limits however the possible capabilities of a model. In a real environment new data are generated frequently, thus it is not possible to rely on a fixed set of data. Mobility data of just one year ago could be very different from the mobility traces generated by the same users in this months of pandemic for example, and this would make the previous model unreliable.

Offline learning thus does not take into account the temporal context such as the period of the year and the year itself, which could have a big impact on the users' movements when considering wide time frames.

Online learning[30] is a different approach which assumes that data arrives continuously, and an up-to-date model is required to have good performances over time.

However, with these assumptions, the data would be too large to be collected and stored in a centralized location, while users' privacy may be exposed to potential risks. A novel approach which promises to solve the online learning issues is *federated learning*, which enables to train and evaluate machine and deep learning models while preserving the privacy of the users.

This thesis focuses on the application of the federated learning paradigm to the problem of the human mobility modelling. First, the thesis provides an in-depth analysis of human mobility models based on deep and machine learning techniques, to then develop and test different applications of federated human mobility models while considering privacy and performance (or utility) aspects.

In the following, we summarize the main contributions of the proposed work.

1.2 Main contributions of the thesis

1. **Design and implement a robust human mobility model.** Design and testing of different server-trained model implementations, such as feed-forward and recurrent neural networks, to develop a good model's design. Furthermore, the implications of building different models for different demographic groups have been investigated. First by developing a synthetic human mobility dataset to overcome the unavailability of real data. Second, building different group models and a population model to evaluate possible improvements in prediction performance.

2. **Development of a federated human mobility model.** This model was to be designed to provide good utility while also protecting the privacy of the users. Privacy protection is thereby achieved by avoiding the necessity to collect and store users' sensitive data on a centralized server.

3. **Investigation of the characteristics and limitations of federated models for human mobility.** The federated models has been analysed in several aspects such as the inherited communication cost, and under different settings such as the sparse availability of the clients; The application of transfer-learning techniques and the influence of differential privacy on the model's utility.

1.3 Thesis Outline

The remainder of this thesis is organized in 4 chapters, which are briefly summarized below.

Chapter 2: Background and Related Work provides an introduction to the core concepts such as human mobility models and federated learning, providing the required knowledge to understand the proposed work. Furthermore, it illustrates the related works alongside with the introduction of the main technologies and concepts such as recurrent neural networks, differential privacy and the fundamentals of federated learning such as the federated averaging process.

Chapter 3: Human Mobility Datasets introduces the three main datasets used in this work. For each dataset is described the pre-processing phases with a description of its features and main characteristics. This chapter also describes the development of the synthetic dataset which has been developed for this thesis and it is so part of the contribution.

Chapter 4: Design and performance analysis of federated learning models for human mobility focuses on the development of a robust human mobility model and the application of the federated learning paradigm. This chapter also presents analysis and results of the experiments conducted to examine the behaviour and the limitations of Federated learning in the context of human mobility.

Chapter 5: Conclusions, Limitations and Future Work discusses and summarises the thesis' results and contributions while providing an overview of the shortcomings, limitations and possible future research lines.

Chapter 2

Background and Related Work

2.1 Introduction

The study of human mobility has gained popularity with the proliferation of location-based services and applications, made possible by the ubiquity of mobile devices. Collection of mobility data is now easier than ever before, every smartphone comes packed with a large set of sensors which can be used to sense the user's movements and collect mobility traces. The availability of human mobility data has enabled the development of human mobility models, which can mimic the underlining patterns that regulate human mobility.

In the last years, several different data-driven approaches have been proposed for human mobility modelling and prediction. Hidden Markov Models (HMM)[31][32], specific probability models for coarse-grained wireless traces (SMM)[33] and, recently, Artificial Neural Networks (ANN) and Recurrent Neural Networks (RNN), have shown promising performances when applied to human mobility[34][3][35]. These methods usually require to collect and store the users' data in a centralized location to operate, exposing the users to privacy risks. The novel federated learning paradigm, alongside with other privacy measures such as differential privacy, promises to develop human mobility models while preserving users' privacy.

This chapter provides an overview of state-of-the-art human mobility models, along with a description of the core concepts such as recurrent neural networks and federated learning. It also presents PMF[10], which, to the best of our knowledge, is the only framework that applies federated learning to the human mobility field.

2.2 Localization of mobile devices

Before entering into the topic, we would like to provide few relevant definitions which will be of main importance in this thesis. Due to the close relation between the movement of a mobile device and its owner, we will consider the location of the device as the location of the user. When describing the movements of a user, it is common to represent them as *mobility traces* or as *mobility trajectories*.

Definition 2.2.1. Mobility Trace. We define the mobility trace of a user u, related to a time interval Δ_t, as the lists the ordered sequence of places visited by the user in that time interval. It can be defined as: $\tau(u, \Delta_t) = \langle p_0, p_1, ..., p_t \rangle$

A more detailed representation of the movements is given by the mobility trajectories which also describe the temporal information of each transition between locations.

Definition 2.2.2. Mobility Trajectory. We define an user's u trajectory Γ as the sequence of places visited by the user at a given day d with the related arrival times and can be defined as: $\Gamma(u,d) = [\langle p_0, t_0\rangle, \langle p_1, t_1\rangle, ..., \langle p_N, t_N\rangle]$

To locate the user, many options are usually available such as GPS, Wi-Fi based localization and cell-towers- based localization. Each technology has its advantages and drawbacks. GPS usually works better under open sky areas, while Wi-Fi based localization provides often a good alternative in indoor environments. However, these technologies can be also combined in order to overcome certain limitations of each. This is true especially on mobile devices which very often provide the full set of sensors, enabling the usage of both GPS and Wi-Fi to increase the localization precision of the device. Geo-location traces use geographic coordinates to define positions on Earth at a given time using mainly the GPS. Locations with a coarser granularity can be acquired instead using different methods, mobile cellular data can define areas given by the transmission range of access points and cell towers where the device is located at some point.

While GPS and cell-towers-based localization are range-based methods, they so rely on the distance between the known transmitters locations and the receiver to compute its location, Wi-Fi based localization is instead range-free. The device sends the fingerprint (Visible WLAN Access points and their information) to a localization service and receives its estimated position. Due to the inaccuracy of GPS signal in the urban environment, the Wi-Fi localization is often used instead to provide more dense data.

2.3 Privacy concerns

With more and more apps collecting and transmitting personal data, users' privacy is in fact more at risk than ever before.

A recent study conducted on approximately a third of the apps available in Google's Play Store by Ram et al. in 2018, has found that on average an app could transfer data to 10 third parties, and one app over five does it with more than 20.

Furthermore, in 2016 V. Taylor et al. analyzed the popularity of permissions across apps present on the Google Play Store. They discovered that the third most popular permission relates to getting the user's fine/coarse location, which represents almost the 25% of the average required permissions.

Data is one of the most valuable resources at the moment, the quote "*Data is the new oil*" has been often use to emphasize how companies have all the interest to collect information to gain knowledge and profit, but the use of those data can be controversial such as in the recent case of Cambridge Analytica[38].

But what is "*Privacy*"? The concept of privacy has developed over time with the introduction of new technologies. One of the more abstract and flexible descriptions is from Warren and Brandeis, 1890 (Harvard Law Review)[39], which described privacy as *"The right to be let alone."* although this definition can cover just a fraction of what privacy means today.

In 1967 a more contemporary and practical definition has been given by Alan Westin: *"The desire of people to choose freely under what circumstances and to what extent they will expose themselves, their attitude and their behaviour to others."*[40]

Westin introduced new aspects that the Warren and Brandeis's definition did not cover. Privacy is not limited to being alone but also concerns the disclosure of personal information which an individual may want to keep private.

How to match the need for data with respect for privacy? Many institutions, such as the European Union, provide strict rules about data management to assure that users are aware of how their data is used.

A possible solution for this problem is often provided by *data anonymization*. Data anonymization is used to unlink the records from the user which has generated them, removing personally identifiable information and providing privacy protection. Anonymization or pseudo-anonymization processes are required by the new European Union's General Data Protection Regulation (GDPR) 2016/679 of 2016[41] which establishes requirements for what constitutes anonymous data, namely: *"... information which does not relate to an identified or identifiable natural person or to personal data rendered anonymous in such a manner that the data subject is not or no longer identifiable."*
Despite their simple definition, completely anonymous data are difficult to achieve and it cannot guarantee that data remain anonymous over time.

De-anonymization can occur when anonymous data from different sources are analyzed and linked together to find correspondences and re-identify the anonymous users.

Just in the last decade, there have been many cases of de-anonymization, including the famous *AOL search data leak*[42].
In 2006, AOL, an internet company, released anonymous search almost 20 Million search keywords searched by 650.000 users where each record contained just an anonymized user-id and the related search keywords. Unfortunately, the anonymized searches did not guarantee user anonymity as expected. A big set of queries performed by the user #4417749 have allowed users to trace it back to the real identity of the user.
Even without identifying users in the report, personal information was however present in many of the search records allowing other users to collect the search history of a user and put together the puzzle. AOL admitted its mistake and retired the data. However, the records were already all over the Internet and so they are still available now on other sites.

Furthermore, an in-depth study on the privacy bounds of human mobility published on Nature[43] has shown that human mobility traces are highly unique. "... *in a dataset where the location of an individual is specified hourly and with a spatial resolution equal to that given by the carrier's antennas, four spatio-temporal points are enough to uniquely identify 95% of the individuals.* "

This explains even better the privacy concerns related to the collection of human mobility traces and highlights the importance of systems which allows the data not to leave the origin.

2.4 Human Mobility Models

The entropy of mobility trajectories has been the subject of a study from Song et al.[20], which has highlighted the high predictability of human mobility and its low entropy. Such characteristics made a potential 93% of human movements highly predictable. In the last decade, several researches have put effort to exploit this predictability with the goal of developing human mobility models able to replicate and predict human movements in specific contexts.

Mobility models are so *"A simplified representation of the movement of single or groups of mobile entities in a given context, primarily the spatial environment"* as defined by Hess et al.

in[21].

Within the large plethora of different mobility models proposed over the years, two major levels of model granularity used to describe movements resolution have been identified[21].

Finest granularity mobility models describe movements in physical space as close as possible to the real world using as granularity the mean precision of the localization technology used. GPS, which provides a global average user range error (URE) of less than 7.8 meters[44], can provide fine-grained traces, such as geographic coordinates.

Aggregated granularity mobility models instead represent the space at an aggregated level, where single close locations are clustered together depending on their distance or other semantic meanings. The granularity of a model depends so by both the availability and the precision of localization technologies and the context in which the model it thought to be used.

In this way, it is possible to simulate or predict the movements of individuals or groups of people at different levels and act accordingly depending on the needed precision or goal. A public transport company could use mobility models to provide additional lines or vehicles on the most common routes, or a network provider could build additional antennas if it is likely that a big group of people would cluster together at some time of the day, such as at train stations, to avoid network congestions.

Trace-based models which use real, anonymous, and large-scale mobility traces data, such as SLAW[45] and SMOOTH[22], produce human walk traces based on statistical patterns and features of real human mobility. SLAW, as SMOOTH, rely on GPS traces of human walks and use those data to extract patterns to then recreate traces which share similar characteristics with real human mobility.

Mobility models for human mobility can be usually classified in 3 different categories[10]:

- Crowd models
- Individual models
- Joint/Group models

Crowd models use data obtained by aggregation of various individuals. The aggregated data does not so provide any reference to the data's owners, reducing the privacy concerns. However, it leaves the risks of the data aggregation phases, in which data of all the individuals are first collected and then aggregated. *Individual models* use instead personal data of the individual users to build a tailored model. The main problems of this latter approach are the sparsity and scarcity of the user's data and the low variance that does not allow a good generalization which can easily lead the model to overfit.

Another problem that Individual models have to face is the so called *cold-start problem*[46][47]. When mobility data of a user has not already been collected or are not enough, such as in the case of a new service just installed on a device, the model is not able to perform its tasks and the user will have to wait until the device collects enough data for the service to work properly.

Recently introduced alternatives are the *Joint models*. Joint/Group models, instead of using data of a single user or considering aggregated data such as crowd models, use data collected from a group of individuals which often tend to share common features (as living in the same city, having the same job or being part same demographic group). This method, however, sets new privacy challenges. Each user has to contribute with its data to the central server's dataset, which will then have access to sensitive data from all the users.

In the last years, many different data-driven approaches have been proposed for human mobility modelling and prediction. Hidden Markov Models (HMM)[31][32] and specific probability

models for coarse-grained wireless traces (SMM)[33] have been used widely by considering location way-points and user activity transitions as observations from which derive probabilities. Recently, artificial neural networks (ANNs) and specifically recurrent neural networks (RNNs), have shown promising performances when applied to the human mobility field as proposed by various publications such as Geo-Temporal Recurrent model (GTR)[34], DeepMove[3] or DPNLP[35], especially with tasks such as the *next-place prediction* [Definition 2.4.1].

Definition 2.4.1. Next-Place prediction. The Next-Place prediction task aims to forecast which place will be visited by an individual (or a group of individuals) after leaving its current place. Where *place* indicates the location of an individual (or group).

One of the main advantages of these deep learning techniques is the easy incorporation of heterogeneous features to better describe and learn the geospatial and temporal context of the movements, alongside with other possible users' characteristics.

2.4.1 Deep Learning

Deep learning is a very successful subfield of machine learning. It has seen quick growth in the last decade due to the increase in computational power and publicly available datasets which have enabled easier access to this research field. The strict relationship with statistics makes it able to identify underlying patterns in data. Deep learning models can deal with large amounts of data otherwise intractable with the common statistical tools. Deep learning has achieved remarkable results in recent years in the fields of computer vision, speech recognition, human mobility modelling and natural language processing.

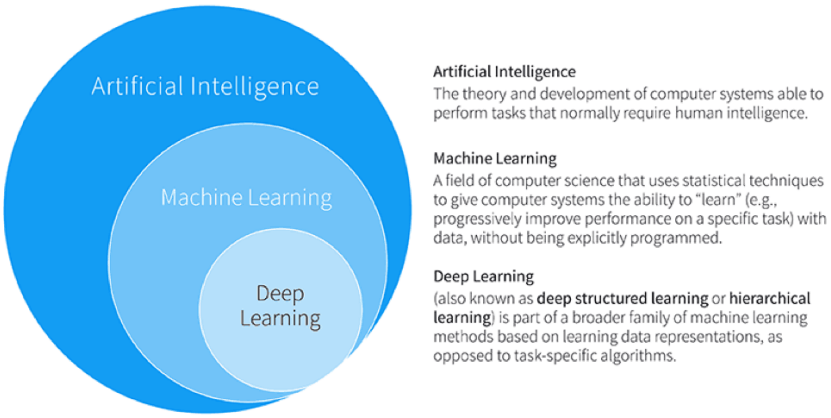


Figure 2.1. The relationship between artificial intelligence, machine learning, and deep learning.[1]

Recurrent Neural Networks

To deal with temporal series of data points, recurrent neural networks (RNNs) are often the first choice for deep learning models. Working on mobility data requires keeping trace of the trajectory history in order to have a better understanding of what will come next. Feed-forward neural networks are not able to do this: Each data point is processed independently and no internal state is kept to keep trace of past locations and transitions. Recurrent neural networks

iterate instead across all the elements of a sequence maintaining an internal state which can represent the previous elements and the spatio-temporal context. The internal state of the RNN is usually resetted before processing another unrelated sequence.

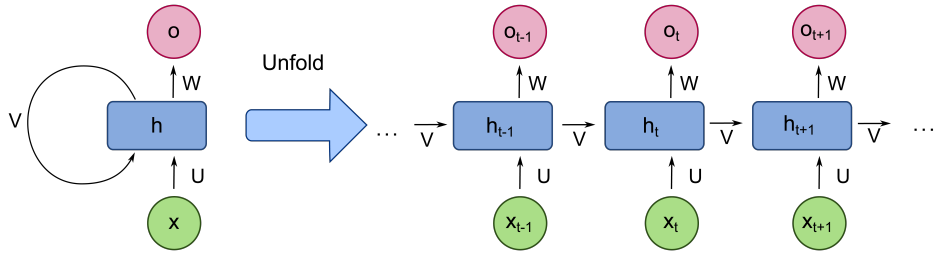


Figure 2.2. Unfolded basic recurrent neural network with only one hidden layer. [2]

Recurrent units are usually Long short-term memory (LSTM) [48] or Gated Recurrent Unit (GRU) [49] which are relatively new but with performance on par with LSTM, while computationally more efficient due to their less complex structure. In the context of human mobility models, RNNs play a central role thanks to the existing close relationship within transition sequences. Human mobility is characterized by complex sequential transition regularities, which can be learned and exploited by RNNs to understand the movement of a user. Despite RNNs can capture regularities within sequences, they fail in the understanding of periodicity. Movements which repeats over large periods are difficult to identify also because of the lack of data representing those events.

The typical mobility data sparsity complicates however the usage of recurrent neural networks. In order for the network to learn a pattern, it needs to see it repeated many times to consider it valuable. Unfortunately, in real scenarios, it is not common to have continuous dense data, and so not all the occurrence of a pattern, which can be a specific sequence of transitions for example, could be captured. Missing locations or transitions could also generate inaccurate data which can lead the network towards wrong assumptions, e.g. a missing place in a mobility trace which could mislead the model.

For these reasons, this thesis will also address the advantages and disadvantages of recurrent neural networks compared with feed-forward networks (section 4.2).

Machine learning and deep learning methods, due to their data-driven approach, usually require a large amount of data stored at a centralized server, which can be difficult to achieve considering the large amount of data that have to be transmitted and stored. Further, these data can contain sensitive and private contents which the user may not want to be collected and transmitted outside its devices (although users often allow to give up part of their privacy to obtain services which can improve their daily life). This can be seen as a tradeoff between utility and privacy. A meaningful example is given by one of the most famous services provided by Google, Google Maps [50], which millions of users allow to collect movement data and let it knows our home and work addresses in exchange for better place and routing suggestions.

DeepMove

Following the encouraging results achieved by deep learning, Feng et al. have proposed DeepMove, an attentional recurrent neural network model for predicting human mobility from lengthy and sparse trajectories which represent the current state-of-the-art for server-trained human mobility models[3].

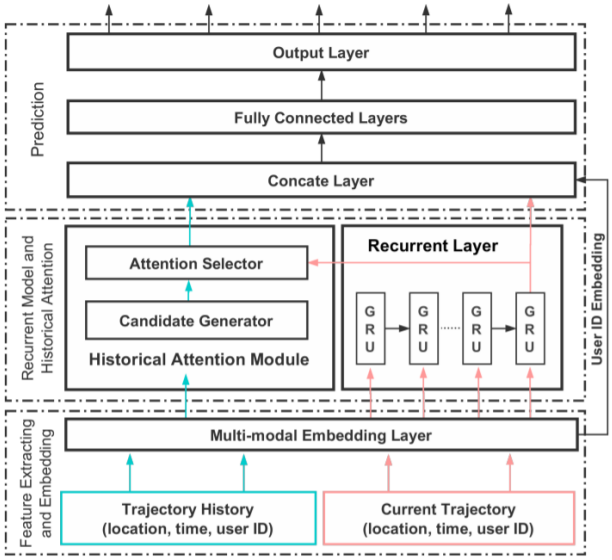


Figure 2.3. Main architecture of DeepMove proposed in [3].

DeepMove is based on a multi-modal recurrent neural network designed with the scope of capturing the various factors that regulate human movements. It relies on an embedding module which converts sparse information (such as the time, the region and the user) into dense representations, then fed into a recurrent neural network to discover transitional regularities. The last module in DeepMove is the historical attention module; its purpose is to capture the periodicity of human movements and select historical records which are correlated with the actual mobility trajectory. This method thus makes use of the user ID information to learn correlations within trajectories and so requires to collect labelled mobility data from all the users, exposing them to potential privacy violations.

2.5 Federated Learning

The need for privacy-oriented machine learning methods has led to the development of federated learning (FL). FL is a novel machine learning method which enables mobile devices (usually called *clients* in the context of FL) to train models without sharing their private data with a centralized server. Data remains so distributed over the large number of clients which have generated them while the model is shared among the server and the devices.

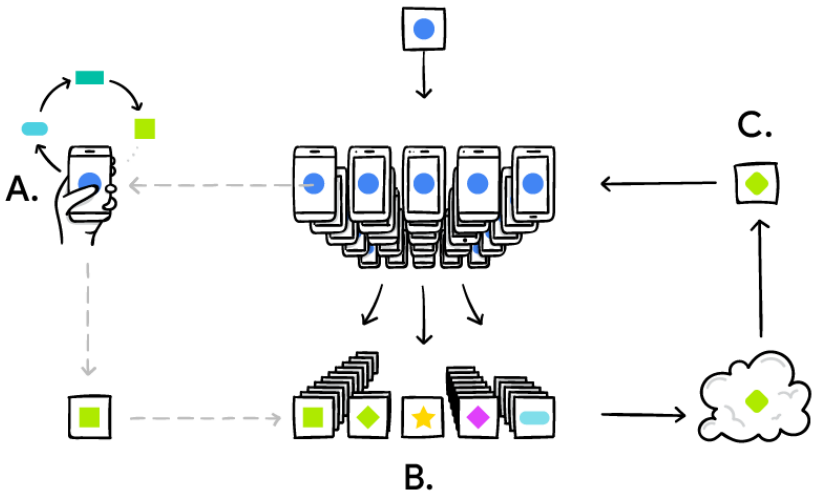


Figure 2.4. Devices personalize the model locally, based on the user's usage (A). Many users' updates are aggregated (B) to form a consensus change (C) to the shared model, after which the procedure is repeated[4].

Federated learning has already been used by Google[4] for its Google Keyboard, a virtual keyboard for touchscreen mobile devices, available on both Android and IOS platforms.

The data typed by users on their smartphones is very often strictly personal and confidential. For this reason, mobile keyboard prediction is a task that fits nicely with the concept of FL, which has demonstrated to work well in situations of unbalanced and not independent or identically distributed (not-IID) datasets, such as text generated by different users.

Federated learning works in rounds[5], where at each round:

- The central server selects a subset of clients that will contribute to the training at this round.
- Each client in the subset downloads the model from the central server and compute several training steps with its local data.
- Once the client has computed the model update, it sends it to the server for aggregation.
- The central server aggregates the updates received from the clients that have been part of the training round, typically using the federated averaging algorithm[8] to improve the model.

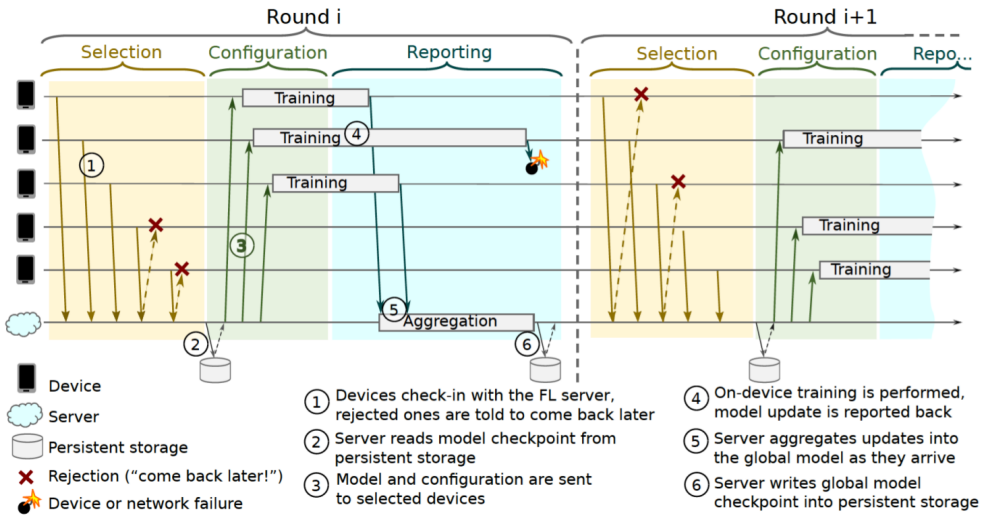


Figure 2.5. Federated learning protocol.[5]

Devices participate to the training round only when they are eligible for the training, i.e., if they have a stable and unmetered Internet connection and they are connected to a power source.

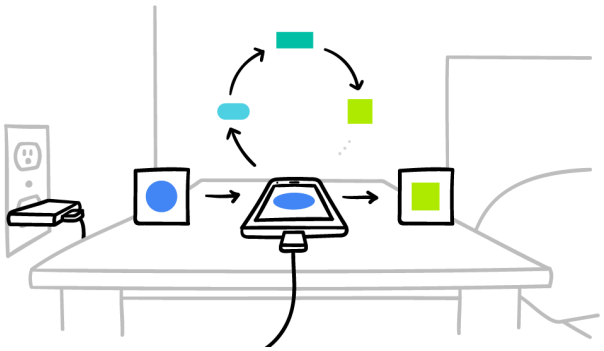


Figure 2.6. Devices participate to the training round only when they are eligible.[6]

The user's data used in the training process never leaves the device, only the model updates do. If an attacker was able to intercept the data exchanged between the device and the central server, or if the server was not honest, it would not see the user's data but it could use the update to extract knowledge about the user who has generated it by reconstructing the information from the update. One method to overcome this security problem is called *secure aggregation*[51]. Secure aggregation ensures that the update is encrypted before leaving the device by adding zero-sum masks to create permutations of the update values. The server will be able to decrypt only the aggregated update, thanks to the zero-sum masks which cancel out once the updates from all the clients get aggregated together so that the server cannot see the results of the single user but only the aggregated data.

Despite the encryption, this is not enough to protect the privacy of the users. Users with

very unique data can generate updates which can expose their contribution or their presence in the training round even after the aggregation of the updates. Here is where *differential privacy* (DP) comes into play. By clipping and adding noise to the model updates it is possible to control how much the model would learn about a single individual, dissipating its private information into long-range statistics[52].

As the training, also the model validation can be done in the decentralized fashion using test data already present on the users' devices. This could be necessary if proxy data are not available on the server and so testing has to be conducted directly on the devices on a subset of the full local datasets.

While at first glance the federated process could look over-complicated to achieve high performance, FL has demonstrated to provide equivalent performances to the common centralized machine and deep learning methods. However, its distributed nature creates new challenges that had not to be taken into account. These include: the communication overhead caused by the exchange of models' updates between the server and the clients or the sparse availability of fast and stable Internet connection of mobile devices which could lead to the failure of training rounds.

In this thesis, we will not deal directly with these problems as we will work in a simulation environment which does not require hundreds of devices but instead just simulate a distributed environment.

2.5.1 Federated Averaging

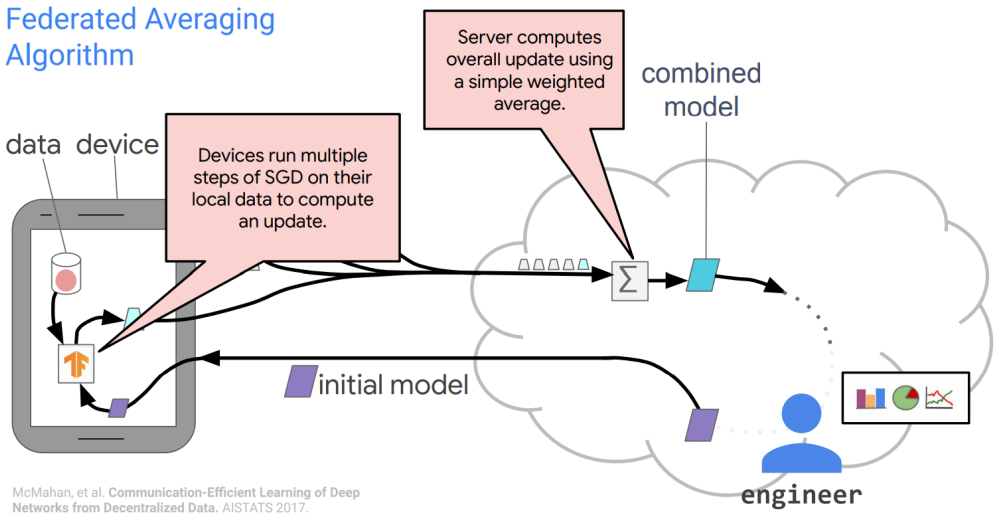


Figure 2.7. Federated averaging example.[7]

The optimization problem, related to FL settings, is called *federated optimization*. Federated optimization has various characteristics that make it different from a typical ML scenario:

- The training data over the devices are non-IID, each user uses different applications and services in different ways and so produces different data.

- The amount of data on each device can be different due to the differences in term of usage of the users.

- The massive number of devices participating in the optimization.

- The sparse availability of fast, unmetered and stable internet connection of mobile devices.

In a real scenario, the federated optimization system should also face and solve various practical problems. Such problems include: the changes in the client datasets whenever new data are added or some are deleted; The sparse and different availability of the clients also considering their environments; Or technical problems with the updates, such as the failure of single updates and so on.

In a common centralized machine learning scenario the computational cost dominates, even with the recent introduction of GPU's computing. In a federated context, the communication cost dominate, with all the common mobile devices problems related to the connectivity.

In a federated optimization environment, clients participate to the training rounds only voluntarily, typically whenever the devices are plugged in charge and under a stable and unmetered Internet connection. Thus, it is plausible to assume that each client would participate only to few rounds per day.

Another consideration should be done relative to the computational cost, since the amount of data per device is expected to be small compared to the complete dataset, the modern smartphones and mobile devices processors should be able to manage the training on these amount of data without requiring long training sessions.

In order to use additional computation to reduce the number of rounds and so, the communication costs, H. Brendan et al.[8] propose two different strategies to add computation:

1. **Increase parallelism** by involving more clients working independently at each round.

2. **Increase the computation** on each client, by adding more local epochs between each communication round.

After having investigated both the approaches, adding more computation on each client achieves the best results in terms of speedups if a base level of parallelism is provided.

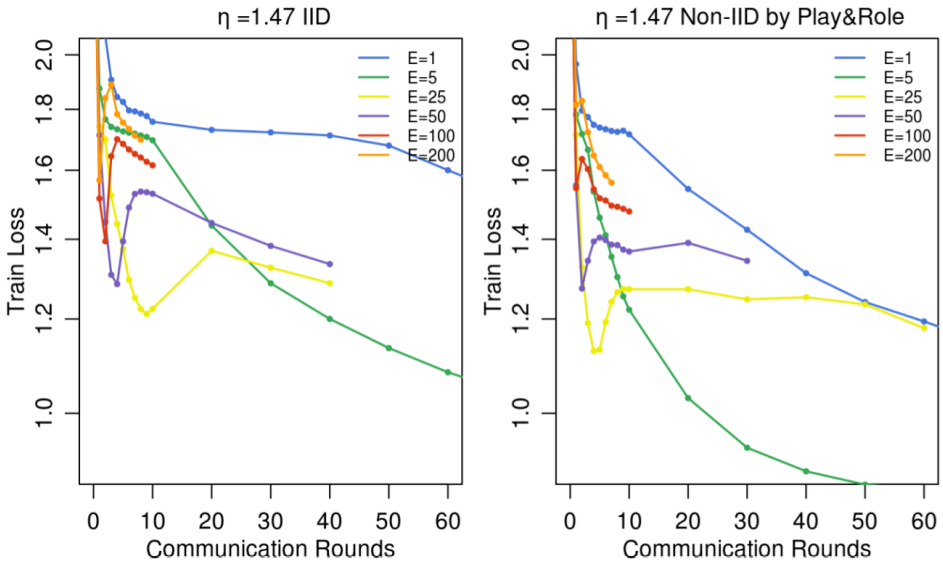


Figure 2.8. The effect of training for many local epochs (large E) between averaging steps, fixing B = 10 and C = 0.1 for the LSTM recurrent neural network using Shakespeare dataset with a fixed client learning rate = 1.47 [8].

To introduce the concept of *federated averaging* is better to first explain *federated stochastic gradient descent* (or FedSGD) which will work as baseline. FedSGD requires to first select a subset C of clients on each round and compute the gradient of the loss over all the data of these clients. Doing so, C represent the global batch size, $C = 1$ denotes the full-batch (non-stochastic) gradient decent

Using FedSGD with $C = 1$ and a fixed learning rate η, each client x would compute the average gradient g_x over its local data at the current model w_t at time t as:

$$g_x = \nabla F_x(w_t) \tag{2.1}$$

The central server, will then aggregate these gradients and update its current model:

$$w_{t+1} \leftarrow w_t - \eta \sum_{x=1}^{X} \frac{n_x}{n} g_x \tag{2.2}$$

Since the summation is defined as:

$$\sum_{x=1}^{X} \frac{n_x}{n} g_x = \nabla f(w_t). \tag{2.3}$$

The definition of FedSGD limits the number of computations per round on the devices, the computation of a new local gradient g_x would require a new communication round.

An equivalent approach is provided by *federated averaging* (or FedAVG). FedAVG updates the current model locally on each device as:

$$\forall x, w_{t+1}^x \leftarrow w_t - \eta g_x \tag{2.4}$$

For then send the new model to the server which performs the weighted average of the resulting models doing:

$$w_{t+1} \leftarrow \sum_{x=1}^{X} \frac{n_x}{n} w_{t+1}^x \tag{2.5}$$

In this way, instead of doing just one step of gradient descent before the averaging step, each device can iterate multiple times across its local data before the aggregation.

FedAVG has three parameters which can be tweaked:

- C, regulates the fraction of clients which participates to a given round.
- E, the number of local epochs each client performs on its local data
- B, the batch size used locally by each client to compute updates.

Using these parameters, FedAVG can also emulate the FedSGD process using $B = \infty$ and $E = 1$. Where $B = \infty$ indicates that the minibatch size on the device is set to the full dataset available, while $E = 1$ makes the clients perform only one iteration over the dataset.

We can then compute the number of local updates per round given the number of local samples n_x as $u_x = E\frac{n_x}{B}$

Algorithm 1 `FederatedAveraging`. The K clients are indexed by k; B is the local minibatch size, E is the number of local epochs, and η is the learning rate.

Server executes:
initialize w_0
for each round $t = 1, 2, \dots$ **do**
  $m \leftarrow \max(C \cdot K, 1)$
  $S_t \leftarrow$ (random set of m clients)
  for each client $k \in S_t$ **in parallel do**
    $w_{t+1}^k \leftarrow \text{ClientUpdate}(k, w_t)$
  $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$

ClientUpdate(k, w): *// Run on client k*
$\mathcal{B} \leftarrow$ (split $\mathcal{P}_k$ into batches of size B)
for each local epoch i from 1 to E **do**
  for batch $b \in \mathcal{B}$ **do**
    $w \leftarrow w - \eta \nabla \ell(w; b)$
return w to server

Figure 2.9. The Federated Averaging pseudocode illustrated in [8].

Algorithm 1, reported in figure 2.13, provides the pseudocode of the FedAVG implementation which has been used for all the experiments conducted by H. Brendan et al. 2017.

It is important to note that in the experiment based on the CIFAR-10 dataset [53], composed of 10 classes of 32x32 images with three RGB channels and 50.000 training samples, not only the

FedAVG algorithm outperform FedSGD, but also an additional baseline, standard SGD training on the full training set (no user partitioning).

While in the evaluation on the test set, the model trained with SGD for 197,500 SGD steps (where each step in the federated settings would correspond to a communication round), FedAvg reaches an accuracy of the 85% on the same test set after only 2,000 communication rounds.

ACC.	80%	82%	85%
SGD	18000 (—)	31000 (—)	99000 (—)
FEDSGD	3750 (4.8×)	6600 (4.7×)	N/A (—)
FEDAVG	280 (64.3×)	630 (49.2×)	2000 (49.5×)

Figure 2.10. Number of rounds and achieved improvement relative to basic SGD as function of target test set accuracy on CIFAR10. [8]

2.6 TensorFlow Federated

The first open-source library to integrate and implement the federated learning settings has been TensorFlow[54] with its TensorFlow Federated (or TFF) open-source framework[55]. TFF has been also used by Google to train its mobile keyboard without uploading sensitive data to the servers.

The TFF framework allows developers to run FL algorithms in a simulated environment using Keras models and centralized data, thus without requiring the availability of hundreds (or even more) of mobile devices which are not easily available in the first research phases in the majority of the cases.

TFF's interfaces are organized in two layers:

- The federated learning API
- The Federated core API

2.6.1 Federate Learning API

The FL API provides high-level interfaces that developers can use to use already existing machine learning models developed with TensorFlow Keras API without changing the model structure or having to re-train the model from scratch. This interoperability allows for instance to use FL to improve an existing model without having to collect again a centralized dataset.

The FL API layer is composed of three main components:

1. **Models.** Classes and helpers which enable the developer to use existing Keras models using wrapping functions or to define custom models to have better control.
2. **Federated Computation Builders.** Helper functions to build federated computations using the models.
3. **Datasets.** Set of data to simulate federated datasets. In a simulation environment, often with research purposes, can be convenient to use data that can be inspected and formatted locally, even if federated learning is designed to be used with data located in a distributed environment where is not possible to have direct access to them.

All these interfaces make use of the Federated Core API that serves as a foundation for federated learning.

2.6.2 Federated Core (FC)

Federated Core (FC) is a development environment that makes it possible to compactly express program logic that combines TensorFlow code with distributed communication operators, such such as those that are used in federated averaging (distributed sums, averages, model broadcasting and so on)[56].

It is important to note that TFF does not use Python under the hood, but an internal language to declare federated computation by a serialized representation. The TFF computations are desired for the system to be deployable not only in simulation environments but also on real physical devices, where Python may not be available.

At the moment, TFF does not come packed with the ability to use differential privacy or secure aggregation, that would be integrated over time. TFF is interoperable with the TensorFlow Privacy library[57] to enable the use of the differential privacy paradigm to train federated models protecting the users' private information; While secure aggregation is provided just by other external libraries such as TF-Encrypted[58] but without integration with the TFF framework, even if it will probably become reality soon.

2.6.3 Alternatives

The possible alternatives to the TensorFlow framework are just a few. The PyTorch implementation of the federated concept is not already complete and well documented as TensorFlow Federated and there is not real official documentation like we have for the TF counterpart. Also, the previously cited TF-Encrypted framework for encrypted deep learning offers something similar to an FL framework but it is also in this case very basic and entirely based on Python, that as we have said previously could be not available on the devices.

2.7 Differential Privacy

Differential privacy (or DP) is a well-known privacy-preserving mechanism for sharing information about groups while withholding information about individuals.

Improper disclosure of sensible and private data could lead to bad consequences. The development of differential privacy helps to mitigate the problem, allowing companies to protect users' sensitive information such as their mobility traces.

> "*Differential privacy (DP) is a strong, mathematical definition of privacy in the context of statistical and machine learning analysis. According to this mathematical definition, DP is a criterion of privacy protection, which many tools for analyzing sensitive personal information have been devised to satisfy.*" *(Nissim et al. in "Differential Privacy: A Primer for a Non-technical Audience"*[9])

From the DP perspective, information data can be seen in two different groups:

1. General information
2. Private information

General information represents any data which does not contain sensitive or identifying content about an individual, while private information is relative to a specific user and can be used to derive knowledge about it.

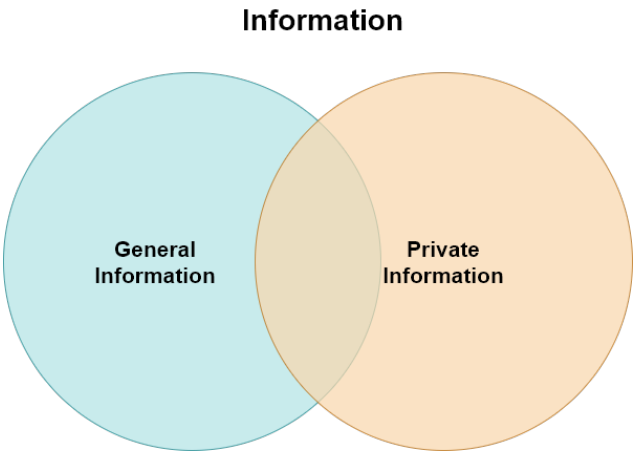


Figure 2.11. Representation of information in data.

Accordingly, the information that can be gained from the data in a differentially private computation includes just what can be learned without including the private information in the computation, with a difference of at most ϵ in the results.

Epsilon (ϵ) denotes the privacy loss parameter, which limits, as upper-bound, and quantifies the individual information which can be learned.

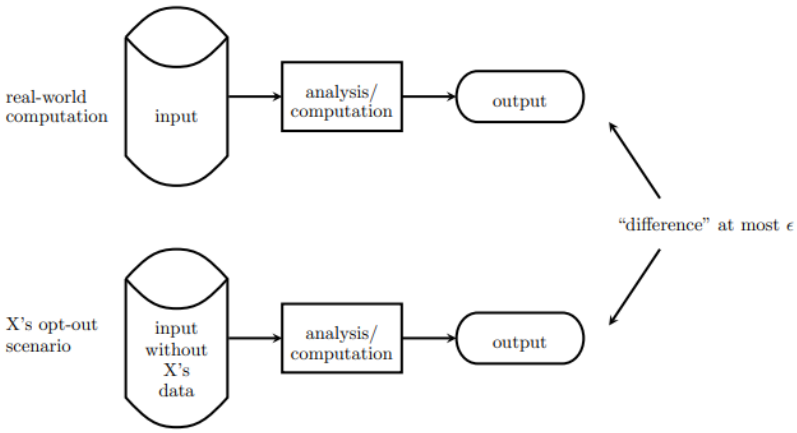


Figure 2.12. Differential privacy. The maximum deviation between the opt-out scenario and real-world computation should hold simultaneously for each individual X whose information is included in the input. [9]

This means that the output of the model should not depend on data of an individual X and removing those data would change the output for at most epsilon. Differential privacy so

guarantees that whether or not the private information of an individual X is part of the dataset used for the differentially private computation, the conclusions drawn from the computation outcome would be the same up to a difference of at most ϵ.

The main weakness of this method is the division in general an private information. DP only protects data which is defined as private information, if something sensible is part of the general group it will not be protected. From the performance (or utility) point of view, the additional noise applied to data can reduce the model's prediction capabilities. It is thus important to reason about the privacy-performance tradeoff when implementing a differentially private model. In particular, Which data need to be protected and which does not.

2.7.1 Differential privacy implementation

One of the latest proposed methods for DP is coming from McMahan et al.[59], which is based on the common Moments Accountant approach of Abadi et al.[60] from 2016 and used in TensorFlow Privacy[57], an open-source library that makes it possible to train machine-learning models with privacy using TensorFlow. The method relies on clipping and adding noise to the records. Records which can represent different things depending on the context (the gradient of one example in case of per-example SGD, the average gradient on one microbatch (10 examples) in case of Microbatch SGD, or model update from one user in case of federated learning settings).

Considering a database X composed by n records, where each record is a vector $x^i \in \mathbb{R}^D$ with the scope of computing the average $\frac{1}{n}\sum_i x^i$. Given a selection probability q, clipping threshold S, and noise multiplier z, the procedure is:

1. A subset of records $R \subseteq [1, ..., n]$, each with a selection probability q, are selected by sampling accordingly with their respective q.
2. Each record x^i for $i \in R$ is clipped to have maximum L_2 norm S by doing $\pi_S(x) = x \cdot min(1, S/\|x\|_2)$.
3. Add Gaussian noise through $\hat{x} = \frac{1}{qn}\left(\sum_{i\in R}\pi_S\left(x^i\right) + N\left(0;\sigma^2 I\right)\right)$ where $\sigma = zS$.

$\hat{x} = \frac{1}{qn}\left(\sum_{i\in R}\pi_S\left(x^i\right) + N\left(0;\sigma^2 I\right)\right)$ represents the output of the Gaussian mechanism for sums. The noise multiplier $z \equiv \sigma/S$ can be used as a knob to to trade off privacy vs. utility. The privacy cost of this mechanism is fully specified by q together with the privacy tuple (S, σ), where S is an upper bound on the L_2 norm of the vectors being summed, and σ is the standard deviation of the noise added to the sum.

2.8 PMF: A Privacy-preserving Human Mobility Prediction Framework via Federated Learning

To the best of our knowledge, the literature provides only one related work describing the use of FL applied to the human mobility field. In March 2020, while this thesis was already started, J. Feng et al. have proposed a *Privacy-preserving Human Mobility Prediction Framework via Federated Learning (or PMF)*[10]. PMF uses the federated learning paradigm to train a deep learning mobility model without requiring the upload of private mobility data to the centralized server, thus only the model parameters are shared. The federated model proposed by J. Feng et

al. is composed by three parts: (1) An input module which perform multi-modal embedding, (2) the recurrent module which makes use of LSTM cells and (3) an output module.

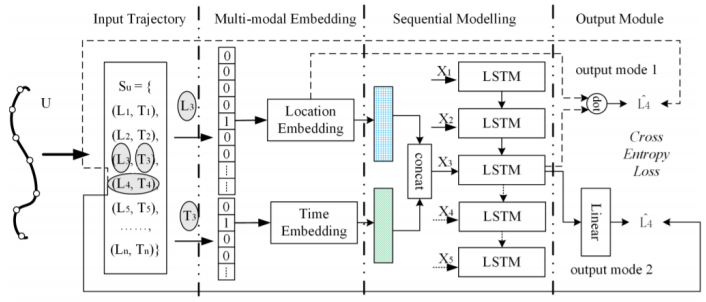


Figure 2.13. The multi-modal model proposed in.[10]

The mobility trajectories used for the training are represented as sequences S_u. Each sequence is composed by tuples of (location, arrival time) which are first encoded using the one-hot encoding vectors and then, two separate trainable embedding layers embed the spatial and temporal information into a dense representation. The concatenation of the two embedded representations of the sequence is then used as input of the LSTM layer.

Following the recurrent module, the output module provides two types of outputs to obtain the probabilities of the possible next locations. The first output module uses a linear layer to increase the dimensionality of the internal state to match the number of possible locations. To obtain probabilities for each location, a Soft-max activation function is then used. The second type of module is a correlation-based method. The internal state is used to compute the correlation between the state and the embedded representation. Also in this case a Soft-max function is used to obtain the probability distribution for the locations. This latter output module has been chosen as the default output for the model due to its better performance.

To further improve the prediction capabilities of the model on the devices, a personal adaptor is used and trained only on the clients' devices for a better personal pattern modelling. The personal adapter, added between the recurrent module and the output module, works as a filter to fine-tune the model with the user's personal information and its most relevant locations.

Although personal data are not uploaded to the server, model updates could be used by attackers or dishonest server managers to mine knowledge about the users. To solve this threat without relying on secure aggregation and extensive use of DP, a privacy-preserving local optimization method is proposed. Their assumption is that while LSTM weights are hard to exploit for an attack, the location embedding module could easily leak some information about the locations visited by the user due to the fact that an update would change only the embedding parameters of the visited places present in the local training data. Modules of the model are so trained differently; while the recurrent module uses normal 'clean' data, the embedding module is trained using noisy data on which DP is applied.

The client-side optimization process of the model uses Stochastic Gradient Descent (SGD) with the proposed privacy-preserving local optimization method; While FedAVG, illustrated previously in section 2.5.1, is utilized for federated learning settings.

Algorithm 1: Training procedure of the proposed system

Global parameters: M^t is the global model at t step; K the number of selected clients; α is the learning rate.
Local parameters: m^k is the local model of client k; p^k is the personal adaptor for client k; *epoch* is the number of local epoch; $\mathcal{D}$ is the private data on the local device; ϵ is the parameter of differential privacy.
Server:
initialize M^0
for *round* $t \in \{1, 2, ...\}$ **do**
  construct/update client candidates pool
  select clients set from candidates pool by polling scheme
  for *client* $k \in \{1, 2, ..., K\}$ ***in parallel*** **do**
    $(m^k, \beta^k) \leftarrow$ **Client**$(M^t, \text{epoch}, \alpha)$
  normalize $\beta^k \in \{1, 2, ..., K\}$
  $M^t \leftarrow \sum_{k=1}^{K} \beta^k m^k$
Client:
// construct noisy data with differential privacy ϵ
$\mathcal{D}_\epsilon \xleftarrow{+noise} \mathcal{D}$
for $i \in \{1, 2, ..., epoch\}$ **do**
  // train the risky module group r of model m with protected data
  $m_r \leftarrow m_r - \alpha \nabla l(m; \mathcal{D}_\epsilon)$
  // train the normal module group n of model m with normal data
  $m_n \leftarrow m_n - \alpha \nabla l(m; \mathcal{D})$
return (m, l)
Fine-tune personal adaptor p **on clients before inference.**

Figure 2.14. The Algorithm 1 illustrates the proposed training procedure. [10]

To perform experiments, three different datasets have been used:

1. **Foursquare dataset**[61]. It is composed of half a million check-ins collected in Tokyo during 10 months between 2012 and 2013. Each record includes an anonymized userID, the timestamp and location information.

2. **Twitter dataset**[62]. It collects over one million geo-tagged tweets created in Los Angeles. Each tweet presents the anonymized user ID, the location, the timestamp and the tweet content.

3. **DenseGPS dataset**[3] The last dataset is from one of the main mobile services provider in China, it contains private data from 5000 users with one month of dense mobility traces. Each record is composed by the user id, the location expressed in geographic coordinates and the relative timestamp.

The locations of the three datasets, represented as geographic coordinates, are projected into 500m x 500m grids to obtain a bigger granularity. PMF has been tested to provide a performance comparison between individual and jointly training settings with the next-place prediction task. The experimental results have shown that joint models achieved better performance when compared with individual models improving the best performance of individual models by 32%. Furthermore, PMF achieved similar results to the state-of-the-art DeepMove [2.4.1], which does not apply any privacy-preserving method, and exhibits promising performances while preserving privacy requirements.

In the following, we summarize the main concepts introduced in this chapter.

2.9 Summary

- **Section 2.2** provides an introduction to the topic and different techniques used to acquire the location of mobile devices.
- **Section 2.4** illustrates the concept of human mobility models and summarises the main approaches present in the literature.
- **Section 2.5** introduces federated learning alongside with its core algorithm, federated averaging.
- **Section 2.6** discusses the choice of TensorFlow Federated as framework and lists the possible alternatives.
- **Section 2.7** focuses on the concept of differential privacy and its implementation.
- **Section 2.8** illustrates PMF, which, to the best of our knowledge, is the only framework that applies federated learning to the human mobility field.

Chapter 3

Human Mobility Datasets

3.1 Introduction

The work proposed in this thesis aims to further explore the possibilities offered by federated learning in the field of human mobility. The spread of the new Coronavirus SARS-CoV-2, and the consequent limitations, denied us the possibility of collecting mobility data directly on the field. Our work makes so use of three relevant mobility datasets to perform the experiments. These datasets include: (1) *NYC taxi dataset*, (2) *Nokia LDCC dataset*, (3) *PERSONA*. While the former two, described in sections 3.2 and 3.4, have been collected and made available by third-parties, the latter one has been developed in the context of this thesis to overcome the initial unavailability of the Nokia LDCC dataset and it's described in section 3.3. This chapter introduces the three datasets and their origins, alongside with the description of the followed pre-processing phases to clean and prepare the data for the usage in the next chapter.

Dataset	City	Duration	Users	Records	Rec./User	Locations	Loc./User	Average density
NYC Taxi	New York City	1 month	13,250	14,283,929	1,078	263	90	50 minutes
PERSONA	-	1 month	60	41,760	696	13	4	1 hour
Nokia LDCC	Lausanne	18 months	185	89,252	482	423	42	16 hours

Table 3.1. Summary of the three mobility datasets.

3.2 New York City Taxi dataset

The NYC Taxi and Limousine Commission (TLC), the agency responsible for managing and licensing the New York City's Medallion taxi cabs (the common yellow taxi) and other types of four-wheels vehicles for the transport of people, provides every year datasets with data related to the licensed vehicles' trips performed during the last year.

These records include attributes capturing pick-up and drop-off datetimes; Pick-up and drop-off locations; Distances of the trips; Trips fares; Payment types; Driver-reported passengers counts and even more attributes.

Although these data represent the movements of taxis and their drivers, they also represent the movements of people who need to move within or even outside the city. Before 2014, the records included two additional fields which are important for simulating a federated learning

approach: The hashes of the taxi cabs' medallions and the hack license of the drivers. While these two information are essential for research purposes in order to subdivide the dataset between the taxis who have generated the records, these fields are no more available in the records provided by TLC. In 2014 Chris Whong, a former engineer, requested and received over 20GB of data featuring more than 173 million trips thanks to a Freedom of Information demand to the TLC.

Unfortunately, there was a problem with the anonymization of the medallion and the hack license fields: they have not been anonymized properly and, with other publicly available data, it was possible to link the taxi's and driver's identities to the related trips.

To anonymized the medallion and the hack license, TLC used MD5, a cryptographic one-way hash function which is supposed to be irreversible. The weakness of the used method is related to the structure of a taxi license number. Taxi licence numbers are composed of 6 or 7-digit numbers (when the first digit is a 5). Due to this construction and other patterns in the structure of these numbers, the possible medallion numbers are just 19 million which made it possible to easily build a rainbow table to de-anonymize all the records with fast modern processors[63] .

While Chris Whong still maintains online a downloadable version of the 2013 dataset[64], since then the two anonymized fields are no more provided by TLC.

For this reason, to have data compatible with federated settings, the original dataset of 2013 has been used for this thesis. The 2013 dataset consists of 12 CSV files, one for each month of the year. The mean number of records for each CSV is around 14 million and, due to the huge dimension of the complete dataset, this work will focus on the dataset with trips recorded in January, which contains data of 14,776,615 trips. Table 3.2 illustrate the fields of each data record.

Feature	**Description**
medallion	Unique taxi cab id number
hack_license	Unique taxi driver license number
vendor_id	A code indicating the TPEP provider that provided the record.
rate_code	The final rate code in effect at the end of the trip.
store_and_fwd_flag	Whether the trip record was held in vehicle memory before sending to the vendor
pickup_datetime	The date and time when the meter was engaged.
dropoff_datetime	The date and time when the meter was disengaged.
passenger_count	The number of passengers in the vehicle.
trip_time_in_secs	Estimated duration of the trip in seconds.
trip_distance	The elapsed trip distance in miles reported by the taximeter
pickup_longitude	Longitude coordinate in which the taximeter was engaged
pickup_latitude	Latitude coordinate in which the taximeter was engaged
dropoff_longitude	Longitude coordinate in which the taximeter was disengaged
dropoff_latitude	Latitude coordinate in which the taximeter was disengaged

Table 3.2. Features of the NYC Taxi Dataset of 2013.

Data cleaning and pre-processing

The large number of rows of the dataset makes it difficult to handle easily using a Python library like Pandas[65], for this reason the records have been imported from the CSV file to a MySQL

local database. The whole thesis has been made using Python as programming language over Colab Python notebooks[66], both with a hosted and local runtimes.

The main advantage of Google Colab Notebooks is that Google provides runtimes hosted for free on powerful virtual machines where high-end GPUs are available, this is fairly important to speed up the training process which could demand a large amount of computational resources that could not be available on the local computers. Also, the TFF framework does not provide already a high-performance environment and the training phase of federated models can require long training sessions.

Since TLC "*makes no representations as to the accuracy of these data.*"[27], it is good practice to first clean the dataset to remove outliers or illegitimate values which may be caused due to some errors. The applied data cleaning procedure consists in three steps: (1) Find and remove null or missing values; (2) Delete records characterized by values which are inconsistent or illegitimate; (3) Remove eventual outliers which can reduce the quality of the data. The dataset provides attributes which will not be useful for the selected task. To further improve the speed of the data manipulation operations and simplify the data cleaning task, these attributes (*'vendor_id', 'rate_code', 'store_and_fwd_flag', 'trip_time_in_secs'*) have been removed.

The data analysis revealed that the attribute '*trip_distance*' presents over 3 million records with null or zero values. This number or records makes this field very unreliable since they represent almost a quarter of the entire records. Because this field will not be part of the features set, those data have not been removed.

A summary of the first data cleaning phase is provided by table 3.3.

Feature	**Null or zero values**
medallion	0
hack_license	0
pickup_datetime	0
dropoff_datetime	0
passenger_count	166
trip_time_in_secs	0
trip_distance	3,399,630
pickup_longitude	285,150
pickup_latitude	285,150
dropoff_longitude	284,897
dropoff_latitude	284,897

Table 3.3. Amount of Null or zero values found for each feature.

The main role played by the *pickup/dropoff_latitude* and *longitude* attributes, makes it impossible to preserve record with null coordinates, thus those records have been removed. The same has been made with records which have null or zero *passenger_count* values due to their limited number.

Once the missing values have been removed, it's necessary to check for possible inconsistencies with the values. The pickup and dropoff coordinates, together with the pickup and dropoff datetimes, are the more important features of the dataset. For this reason, they have to be controlled carefully to avoid further problems during training.

Pickup and dropoff coordinates can present two kinds of possible inconsistencies: (1) The

two coordinates are identical; (2) The coordinates are out-of-range.

Although it's possible that a taxi trip starts and end in the same location, it's also very unlikely. The analysis shows the presence of more than 400,000 trips which ends exactly when they started, these records have been considered faulty and so removed from the dataset.

Geographic coordinates, expressed in terms of latitude and longitude, can vary from -180°and 180°, extremes included. However, since the dataset is related to trips performed by taxis of New York City, not the whole range of values can be valid. It is necessary so to define a range, defined as a subset of [-180°, 180°], of values in which the location coordinates of a trip can be considered feasible. New York City is situated in the state of New York , and it borders on the states of New Jersey, Connecticut and Pennsylvania. Since it is possible that a passenger could need to go to one of the surrounding cities closest to the state of New York, a "*bounding box*" comprising the four states has been defined to includes the four states. A public repository of Jack Bathman on GitHub[67] provides the approximate maximum and minimum latitude and longitude for all states and so defines the single bounding boxes.

		Lower Bound	Upper Bound
NY	Longitude	-79.7624	-71.7517
	Latitude	40.4772	45.0153
NJ	Longitude	-75.5708	-73.8885
	Latitude	38.8472	41.3593
CT	Longitude	-73.7272	-71.7874
	Latitude	40.9509	42.0511
PA	Longitude	-80.5243	-74.707
	Latitude	39.7199	42.5167

Table 3.4. States bounding boxes.

Using these values, by taking as a minimum the global minimum and as maximum the maximum, it is possible to define the range of coordinates in which all the four states are bounded in. Table 3.5 defines the valid area.

	minimum	maximum
Longitude	-80.5243	-71.7517
Latitude	38.8472	45.0153

Table 3.5. Valid area bounding box.

The analysis of the records showed that there were trips way out of range with latitudes and longitudes greater than 2000°. All the records with pickup or dropoff locations outside the valid area just defined have been considered faulty and removed.

Further inconsistencies have been found looking at the pickup and dropoff datetimes. While 34,308 records present identical pickup and dropoff datetimes, few rows have a pickup datetime greater than the dropoff. The last problem found was related to the *passenger_count* field which reported trips with over 255 passengers. The maximum number of passenger that has been

considered valid is 6.

Overall, more than 460,000 records have been discarded because afflicted from one or more of the cited problems; leaving 14,315,662 valid records.

To further clean the data, three additional derived features have been computed starting by the valid records fields *trip_distance*; *trip_duration* and *trip_speed*.

While the duration of a trip is trivial to obtain by the difference of the dropoff and pickup datetimes; the trip distance requires the introduction of the concept of *Haversine distance* to be computed. The Haversine distance between two points, as well described by the SKlearn python library which offers an implementation of it[68], measures the angular distance that exists between two points on the surface of a sphere. Thanks to the fact that the Earth is nearly spherical (Checkmate flat Earth society), the Haversine formula well approximate the distance between two point located on the Earth's surface with an average error of less than 1%. Equation 3.1 shows the formula to compute the Haversine distance in kilometers.

$$D(x,y) = 2\arcsin[\sqrt{\sin^2((x1-y1)/2) + \cos(x1)\cos(y1)\sin^2((x2-y2)/2)}] * 6371 \quad (3.1)$$

Since in reality, especially in large cities, it is not possible to move in a straight line; The Haversine distance of a trip would be far smaller than the real distance travelled by the taxi. So also the estimated trip speed based on the Haversine distance would be smaller than the real one.

Once the trip time and the trip distance are calculated, the estimated speed can be computed simply as $trip\_speed = \frac{trip\_distance}{trip\_time}$

The first implementation of the code to compute these attributes used the Pandas "*apply()*" function which applied a given lambda function to every row of the pandas dataframe. The problem with this method is that the apply function returns a Pandas Series for each row on which the lambda function is invoked and so the execution takes many minutes. The way to solve this issue is by using vectorization. Thanks to the optimization of Numpy arrays, it was possible to reduce the computational time from ten minutes to just a bunch of seconds simply by working with dataframe's columns as Numpy arrays instead of relying on the pandas apply function.

Table 3.6 shows the minimum, mean and maximum values of the computed additional features in the dataset.

Features	minimum	maximum	mean
trip_distance	0.0 km	539.3 km	3.428 km
trip_time	0.005 h	51.015 h	0.191 h
trip_speed	0.0 km/h	216.7 km/h	16.7 km/h

Table 3.6. Minimum, mean and maximum values of the additional features.

All the three added features present abnormal ranges of values. To overcome this problem, as the last step of the data cleaning phase, criteria have been defined to identify possible outliers. The statistic provides two common methods to identify outliers, The *Z-score* and *IQR - interquartile range*[69]. While the Z-score relies on the mean and standard deviation of the samples to assign a score to the data points which represent how far are them from the overall

distribution; IQR use the first and the third quantile to tell when a data point is too far from the middle.

Both these methods are however limited because they can't make use of knowledge of the context in which data are generated. An example is given by the Z-score computed over the *trip_speed* feature which identifies as upper bound a speed of 83.7 km/h, that as said before, would correspond to greater real speed. The Los Angeles Times reported in 2018 that the average speed in midtown Manhattan was around 4.7mph (7.5 km/h)[70] and other sources cited 15 km/h as a reasonable average speed for NYC, considering 50km/h as a possible realistic limit[71]. All these values are far from the speed estimated by Z-score, and also looking at trips with a duration over 20 hours they are mainly characterized by speeds close to 0. All these considerations have contributed to the decision of the validity ranges here reported.

Feature	minimum	maximum
trip_distance	0.1 km	70 km
trip_time	0.1 h	4.0 h
trip_speed	5 km/h	60 km/h

Table 3.7. Validity ranges for the additional features.

After having removed the outliers, the number of valid records has decreased to 10,611,605.

Commonly, a taxis have to move to the next pickup location in between two consecutive taxi trips. And this can cause the dropoff location of a trip not coinciding with the pickup location of the following one. These movements with no passengers however are not represented in the dataset. To further increase the density of the mobility data, intermediate "hidden" trips have been computed and integrated with the cleaned dataset.

Definition 3.2.1. Intermediate Trip. Considering two different but consecutive trips A and B, where A started from the pickup location P_A and ended in the dropoff location D_A; B started from the pickup location P_B and ended in the dropoff location D_B. The intermediate trip C will have as pickup location P_C the dropoff location of A (D_A) and as dropoff location the pickup location of B (D_B). C will exists if and only if the medallion of the taxi and the hack license of the driver which performed A and B coincide; the distance between D_A and P_B is in a valid range and the time between A and B fall in a defined range of values which makes it feasible.

Feature	minimum	maximum
trip_distance	0.5 km	20 km
trip_time	0.1 h	1.0 h
trip_speed	5 km/h	40 km/h

Table 3.8. Intermediate trips validity ranges for the additional features.

The validity constraints for intermediate trips are defined in table 3.8.

The intermediate trips have been computed by considering pairs of consecutive records and calculating the additional three feature as done for the original records. The total number of records available after the addition of the intermediate trips is equal to 14,283,929.

With all the cleaned records in place, a data pre-processing and features engineering stage has been required to properly use the dataset with deep learning techniques. Pickup and dropoff datetimes include too many temporal information to be well interpreted by a neural network. These features, in the format of "yyyy:mm:dd hh:mm:ss" have been decomposed into three more meaningful attributes: (1) The day of the week [1-7]; (2) The hour of the day [00-23] and (3) the month [1-12].

The major concern is instead related to the representation of pickup and dropoff locations. Geographic coordinates expressed in terms of latitude and longitude are too fine-grained to be learned by a neural network and provide proper results. TLC in the latest years has already moved from the latitude-longitude representation to a more coarse-grained view of the locations as Taxi-zones and boroughs.

The New York City Open Data, a hub of analytics in the City of NY, provides the spatial description of each taxi zone of NYC[72]. The zones are based on NYC Department of City Planning's Neighborhood Tabulation Areas (NTAs) and approximate neighborhoods of the city.

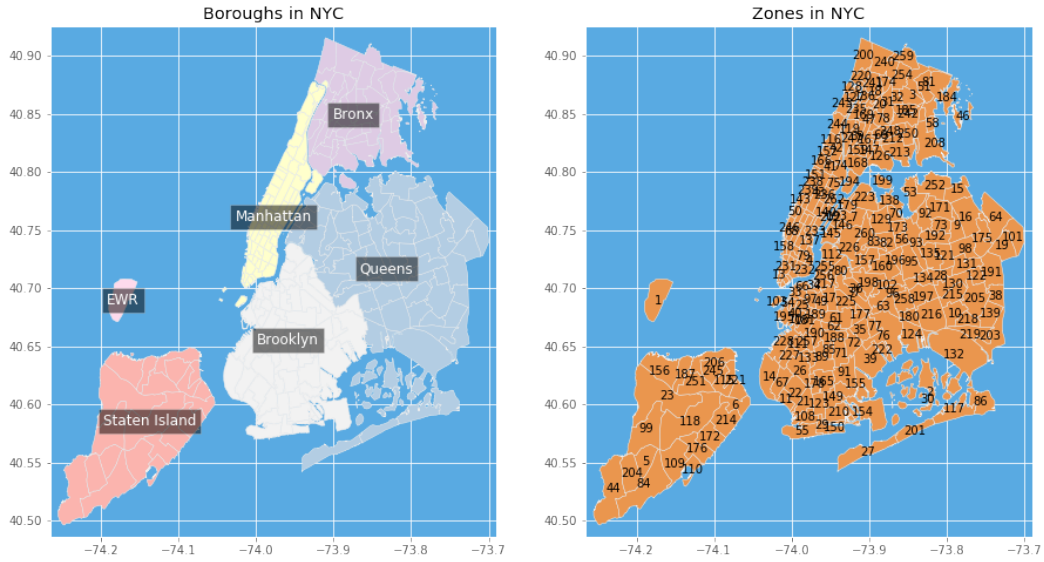


Figure 3.1. An overview of the taxi zones of NYC provided by Chih-Ling Hsu.[11]

The number of zones in NYC is 262, each zone is defined as a MULTIPOLYGON which delimits the area of the zone. A MULTIPOLYGON is an object that represents a collection of POLYGONs, where a polygon is a sequence of coordinates which define the vertices (or POINTs) of the polygon's area. The problem of mapping the locations defined by latitude and longitude coordinates can be solved by finding the polygon (the taxi zone) which contains the point (the location defined by the two coordinates). To solve this problem, two Python libraries have been used: (1) *PythonGIS* and (2) *GeoPandas*. PythonGIS allows to define and manipulate objects of type POINT, POLYGON and MULTIPOLYGON, while GeoPandas extends the common pandas library to allow spatial operations on geometric types.

Because in the dataset there are locations which are not located in NYC due to how the valid area bounding box has been defined; All the pickup and dropoff locations situated outside NYC do not match with any taxi zone in the spatial join operation, that allows joining POINT

objects (as the pickup and dropoff locations) with the POLYGON or MULTIPOLYGON object that contains them (the taxi zone). All these locations have been mapped manually to an abstract extra taxi zone which represents the surroundings of NYC.

The final pre-processed dataset counts 14,283,929 trips characterized by the features illustrated in table 3.9.

Name	Description	Values range
medallion	Unique taxi cab id number.	-
pickup_weekday	Day of the week when the meter was engaged.	[1, 7]
pickup_hour	Hour of the day when the meter was engaged.	[0, 23]
pickup_day	Day of the month when the meter was engaged.	[1, 31]
pickup_month	Month of the year when the meter was engaged.	[1, 12]
pickup_location_id	Taxi zone where the meter was engaged.	[0, 262]
dropoff_weekday	Day of the week when the meter was disengaged.	[1, 7]
dropoff_hour	Hour of the day when the meter was disengaged.	[0, 23]
dropoff_day	Day of the month when the meter was disengaged.	[1, 31]
dropoff_month	Month of the year when the meter was disengaged.	[1, 12]
dropoff_location_id	Taxi zone where the meter was disengaged.	[0, 262]

Table 3.9. Final pre-processed NYC Taxi dataset schema.

3.3 PERSONA synthetic dataset

This dataset born from the need of mobility traces provided with demographics data of the users. Baumann et al. in "*Selecting Individual and Population Models for Predicting Human Mobility*"[73], after several experiments related to mobility models for demographic groups, concluded that only a little gain in performance can be achieved by developing different mobility models for different demographic groups respect to a shared population model built with data of all the users.

To further explore these results, due to the absence of demographic data in the NYC taxi dataset [section 3.2] and the initial unavailability of the Nokia LDCC dataset [section 3.4], it has been required to develop a synthetic dataset with the proper characteristics to derive and compare groups models against a population model.

PERSONA has three main demographic groups (or main classes): (1) the student class; (2) the employee class and (3) the retired class.

Each main class has six possible relevant places and three subclasses which are specializations of the main class. Additionally, an extra generic place *'other'* is provided to all the classes to represent any other possible relevant place which could not be identified or known.

Each class specialization has a subset of the relevant places of the parent class. This allows different subclasses to have similar but not identical relevant places, augmenting the variance and the realism of the generated mobility traces.

		Home	School	Park	University	Library	Station	Work	Parents' House	Grocery Store	Gym	Restaurant	Church
Student	Type A	X	X	X	-	-	-	-	-	-	-	-	-
	Type B	X	-	X	X	X	-	-	-	-	-	-	-
	Type C	X	-	-	X	-	X	-	-	-	-	-	-
Employee	Type A	X	-	-	-	-	-	X	X	X	-	-	-
	Type B	X	-	-	-	-	X	X	-	-	X	-	-
	Type C	X	-	-	-	-	-	-	X	-	X	-	-
Retired	Type A	X	-	X	-	-	-	-	-	X	-	X	-
	Type B	X	-	-	-	-	-	-	-	X	-	X	X
	Type C	X	-	X	-	-	-	-	-	-	X	X	-

Table 3.10. Main classes, sub-classes and their relevant places.

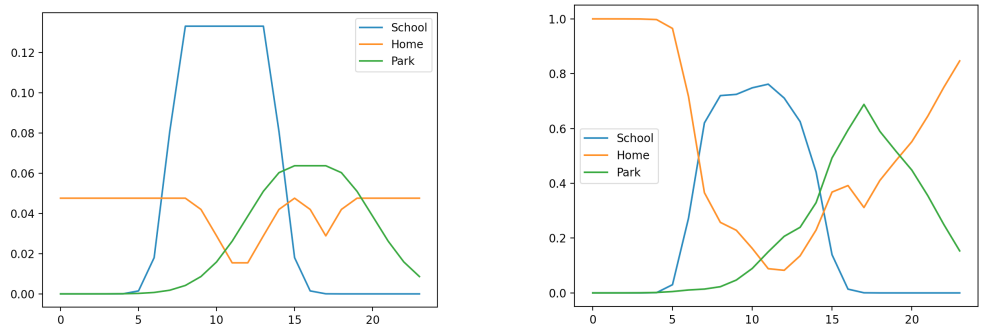
An example can be given by the student class, while master's and high school's students fit both in the student category, their movements and relevant places can very easily differ. While is more likely for a master's student to come to its university from an another city by train, a teenager in the high schools probably can reach its school by feet or by short-range public transports like bike-sharing or buses. Each subclass is defined simply as a Python dictionary. For this reason, new classes and subclasses can be defined very easily by adding or editing elements in the dictionary. To generate mobility traces, each place should have, similarly to what states have in Markov Models, a starting probability and transition probabilities from and towards any place for each hour of the day; Possibly with differences between week and weekend days. The proper way to represent these probabilities is with matrices where each subclass have a matrix with starting place probabilities which describes for each relevant places of the subclass, the probability to start the day in that place. Additionally, place transition probabilities would require three-dimensional matrices to be described taking into account the fact that these probabilities would depend not only on the current place but also on the time of the day. While the starting place can be easily fixed to a default location such as *Home*; Overall this system would require the definition, considering an average of 4 places for each one of the 9 subclasses, of (((4 places x 4 places x 24 hours) x 9 users values for the transition probabilities matrices. For a total number of 3,456 values to decide by hand. This is clearly infeasible. In order to overcome this definition problem, PERSONA uses an approximation of the probabilities matrices.
The solution is simple but effective, instead of defining probabilities for each hour of the day, different Gaussians ditributions are fitted in the main hours of each place. Each place has hours in which is more likely for a user to be, as for a student is more likely to be at school at 10 while is very improbable for him to be there at 23. Each subclass has so a set of hours in which is highly probable to be in a specific place, this set of hours are used as means for several normal distributions. While the mean specifics the hot-hours for that location, the standard deviation specifies instead how well defined that hours are and how much those hot-hours can vary.

The different Gaussians defined for each users' subclass and each place are then combined to obtain a unique probability value for each hour of the day. To merge overlapping probabilities three methods have been considered: (1) take the maximum probability, (2) take the minimum probability or (3) take the average probability. From different tests, taking the maximum provided resulting distributions closer to the desired one.
These probabilities can describe only one place at time, to generate movements between different places transition probabilities are required. This requires an additional function, normalize. The scope of normalizing is to make the sum of all the probabilities of different places at the same hour time slot equal to the desired probability value P.

Because probability distributions of places are defined individually, they could present dif-

ferent scales and the sum of the probabilities of different places can be either close to zero or larger than 1.0.

The normalization function operates on all the places for each time slot and solves the scale problem, an example of the result of the application of the function is given by figure 3.2.



(a) Probability distributions pre-normalization

(b) Probability distributions post-normalization

Figure 3.2. Comparison between the probability distributions of three different place before and after normalization.

Probabilities are stored into a two-dimensional array where for each hour of the day there is a list with a probability value for each place in that time slot. Since each entry of the generated mobility trace is contained in a time of fixed length (one hour in this case), the mobility trace is called *'Slotted'*.

Definition 3.3.1. Slotted Mobility Trace. We define the mobility trace Ω_s of a user u related to a time interval Δ_t lists the ordered sequence of places visited by the user in that time interval by considering time windows of length s. It can be defined as: $\Omega_s(u, \Delta_t) = \langle p_{(0,s]}, p_{(s,2s]}, \ldots, p_{(t*s,(t+1)*s]} \rangle$

The location of the user at a given time t is then simply random sampled from the correspondent list of place probabilities. To further replicate realistic human mobility, a stability value is used to increase the probability of the "*stay-in-place*" action, making more likely for a user to stay in the current place instead of change location frequently.

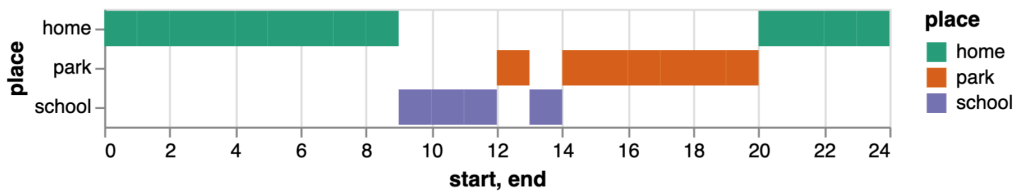


Figure 3.3. Simulation of a synthetic slotted mobility trace produced by a student.

Very often, routines of people change between week and weekend days. Students may not go to school or to the university in the weekends, and so the probability of visit those places

should be very low in those days of the week. To implement this distinction, each place's dictionary has an additional field which specifies if the location is visited only during the week, only during the weekend or in both cases. Whether a weekday is simulated, a place which usually is visited only during the weekends gets as normalization threshold a maximum probability of 0.1 that makes it very unlikely to be sampled.

The generation of the mobility traces is simple, each user's subclass has the same probability to be sampled.

Algorithm 1 PERSONA dataset generation

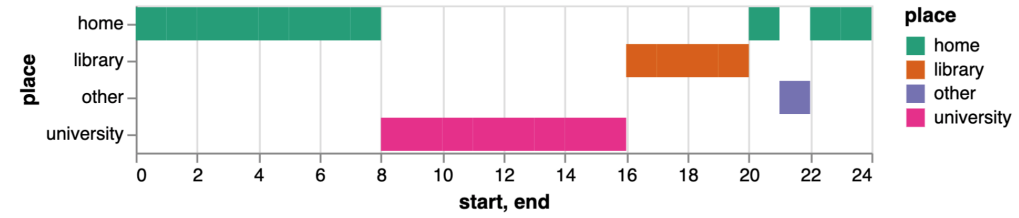
function GENERATE_TRACES(D, N) ▷ Generated D days of traces for N users
  Generate N users by randomly select between the defined subclasses
  $dataframe = DataFrame()$
  for $day = 0, 1, 2, \ldots, D-1$ **do**
    for $user = 0, 1, 2, \ldots, N-1$ **do**
      $p_{hour} \leftarrow' Home'$ ▷ Start from Home
      $trajectory \leftarrow [p_{hour}]$
      for $hour = 1, 2, \ldots, 23$ **do**
        $p_{hour} \leftarrow sampleFromPlaces(user, day, hour, p_{hour})$
        $trajectory.append(p_{hour})$ ▷ Add the sampled place to the trajectory
      end for
      $dataframe.add(trajectory)$
    end for
  end for
  return $dataframe$
end function

N subclasses of users are selected randomly. For each day of the D days that needs to be generated, 24 places (one per hour) are sampled from the probability distributions for each of the N users. Each record is stored together with other information such as the userID; The user's class; The time and the previous location, into a pandas dataframe which has the schema illustrated in table 3.11.

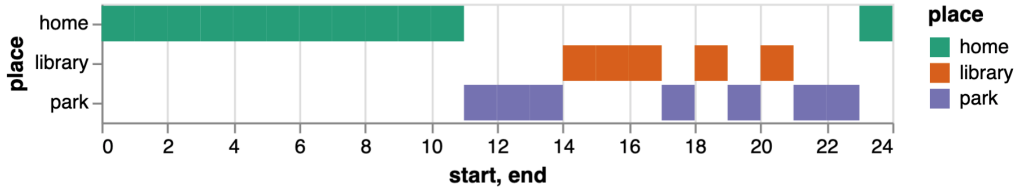
Name	Description	Values range
users_id	Unique user's id	[0, N-1]
users_class	Main class type of the user	[0, 2]
start_hour	Hour of the day when the user is at start_place	[0, 23]
end_hour	Hour of the day when the user is at end_place	[0, 23]
start_place	Initial relevant place of the user	[0, 12]
end_place	Final relevant place of the user	[0, 12]
day_type	Whether is a weekday or a weekend	[0, 1]

Table 3.11. Final pre-processed PERSONA dataset schema.

Figures 3.4, 3.5 and 3.6 provide an example for both week and weekend days for each user class generated.

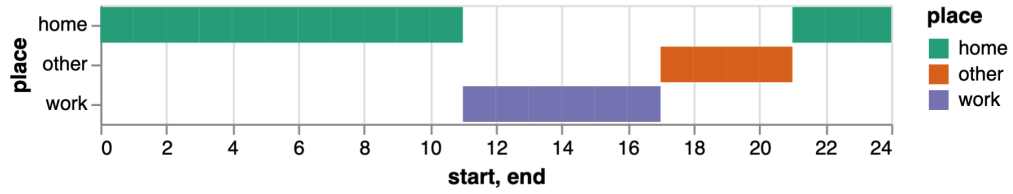


(a) Weekday

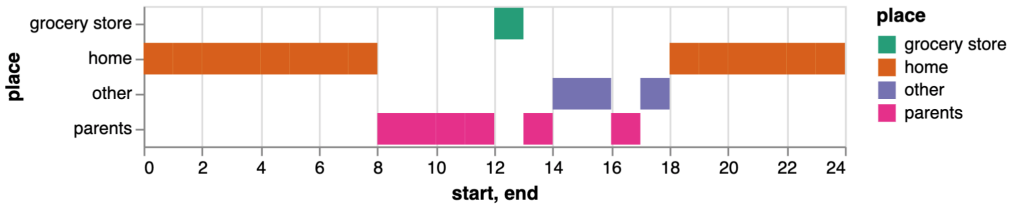


(b) Weekend

Figure 3.4. Mobility traces generated for an university's student in a week and a weekend day.

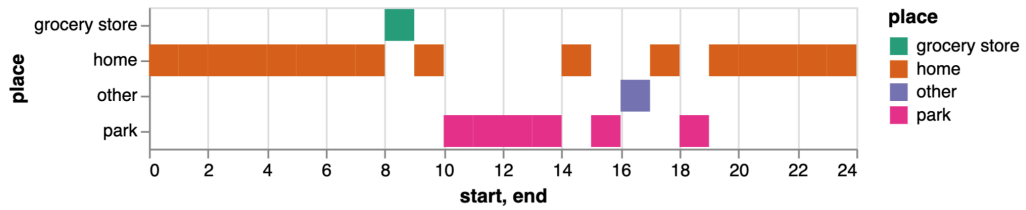


(a) Weekday

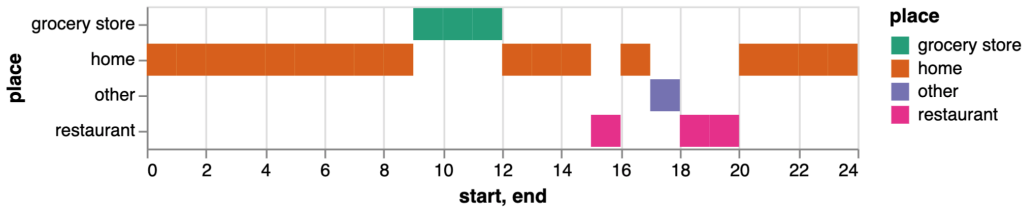


(b) Weekend

Figure 3.5. Mobility traces generated for an employee in a week and a weekend day.



(a) Weekday



(b) Weekend

Figure 3.6. Mobility traces generated for a retired in a week and a weekend day.

The final dataset generated with PERSONA comprehends 30 days of records of 60 unique users.

3.4 Nokia LDCC dataset

In 2009, the Nokia Research Center in Lausanne started a data collection campaign with the goal of creating a large-scale mobile dataset[74]. In this context, smartphone data of 191 volunteers have been collected during a period of almost 18 months in the Lake Geneva region using Nokia N95 mobile devices. The data present in the LDCC dataset includes, but it is not limited to, WLAN and Bluetooth scans, GPS traces, Demographics data of the users, visible cell towers and phone call logs.

The demographic data includes information about 155 of the 191 users, including the gender, the age group and the current occupation. Because the collected data includes a considerable amount of sensitive information, such as the user locations and phone numbers, data anonymization techniques have been applied to protect the users' privacy. Such methods include the truncation of position coordinates and the hashing of phone numbers and WLAN network identifiers such as MAC addresses.

Location related data include GPS traces and WLAN scans. While WLAN scans data can be used to infer the users' locations, the accuracy provided is often quite low. Because the fraction of time in which the participants have been able to collect GPS data is only the 2.4%, the two methods have so been used together. The dataset provides the location of 57% of the WLAN access points, computed by combining GPS data and WLAN scans. This allows the localization of the users for the 61% of the time, even if the accuracy of this methodology heavily depends on the density and the distribution of the access points. Due to the limitations of the software used for the data collection, data of only one cell tower has been reported at each time, precluding any possible triangulation technique. Unfortunately, due to the impact of the global pandemic

and nasty bureaucratic procedures, the availability of this dataset has been delayed until the begin on August, limiting its possible applications in the development of this thesis.

Data cleaning and pre-processing

The dataset is composed of 31 distinct CSV files, each of which is related to a specific type of data. The first CSV that has been considered is the so called *visit_10min*. While at first glance the *visit_10min* table, that reports the history of place visits by the participants (on 10 minutes bases), was suitable to be used similarly as done with the PERSONA dataset, a deeper analysis of the data revealed that almost the 95% of the places are not labelled neither with the related location coordinates nor with their semantic meaning (such as *Home*, *Work* and so on). These reasons have made the usage of this data impossible for the purposes of the thesis.

The WLAN scans are instead subdivided into three different files, namely *wlanrelation*, *wnetworks* and *gpswlan*, which have to be joined together to obtain all the information related to the observed WLAN access points. The first file (or table) contains the *db_key* of the observation, which represents the primary key for all the records in the dataset and the WLAN access point's SSID with an unique id. By joining the first two tables it is possible to obtain the MAC address of the observed access point, which can then be used to obtain the geographic coordinates of the AP contained into the latter table. Through the *db_key* attribute is then possible to link each record to the related user, specified by an unique *user_id*. Each user's identity is described by its unique phone number and its demographic information.

Contrary to what has been claimed in [74], just slightly more than 10,000 over almost 600,000 access points are provided with their geographic coordinates, which represents only the 1.6% of the total access points. The number of WLAN scans provided is more than 53 million, but due to the scarcity of the access points locations, the number of usable records is reduced to just 332 thousand. Furthermore, due to the truncation of position coordinates used to protect the privacy of the users, many WLAN access points' locations present low accuracy. While coordinates with a number of decimals over 3 can provide enough precision for our needs [75], access points with a lower location accuracy had to be removed. The final pre-processed dataset of WLAN scans counts 158,083 records.

In addition to the WLAN records, the GPS traces have been used to increment the number of records. Unlike WLAN scans, GPS traces are both sparse and dense in time. When the user is under an open sky, the records are very dense with almost one record every 10 seconds. When instead the user is in an indoor environment, there can be no records many hours. In order to combine GPS and WLAN traces without compromising the quality of the data, the GPS traces acquired outside the *Lausanne's area* have been removed and the minimum time between two consecutive locations, for both the WLAN and the GPS observations, has been set to 30 minutes by removing the extra data points.

The Lausanne's area has been defined as a bounding-box as follow:

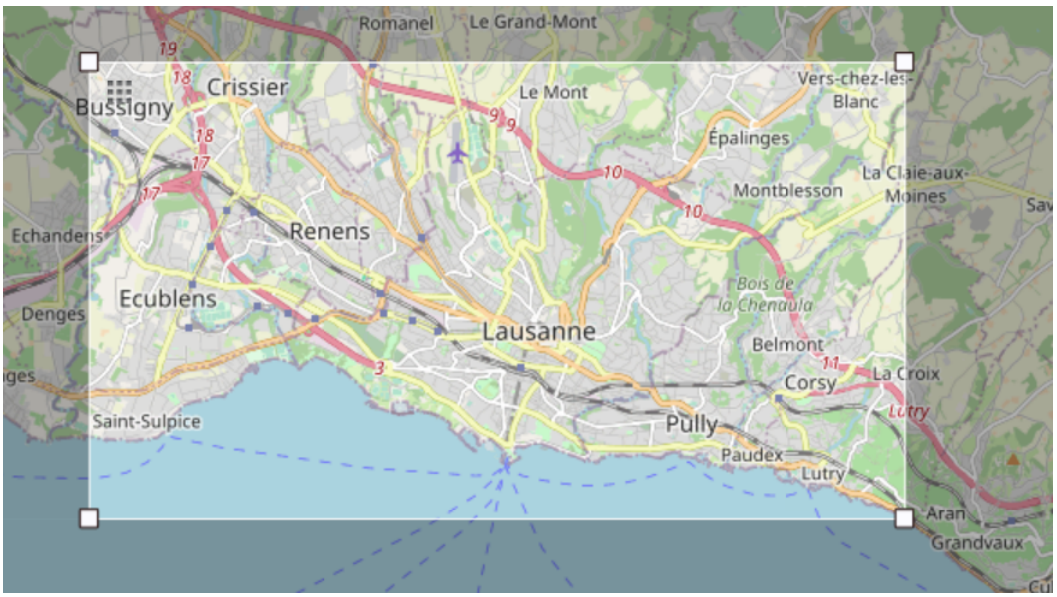


Figure 3.7. The bounding box which defines the Lausanne area.

	Lower bound	Upper bound
Longitude	6.5472	6.7023
Latitude	46.4972	46.5570

Table 3.12. Bounds of the bounding-box defining the are of Lausanne.

The dataset originated by the union of GPS and WLAN traces counts slightly less than 90 thousand rows generated by 185 unique users, 153 of which have demographic data provided. The median number of data points for each user is 376, the average number of records per day is instead only 3.15, while the average number of days per user is 145.2 with a number of users with more than 30 days of records of 154.

The dataset locations expressed in terms of latitude and longitude are too fine-grained to be learned by a neural network and provide proper results. Similarly as done by New York City Open Data, the Swiss confederation provides a shapefile with the boundaries of cantons and municipals[76]. The borders of each Swiss canton are defined by a sequence of POLYGON objects which define the shape of the area. The procedure followed to join geographic coordinates with the related cantons is the same explained in section 3.2.

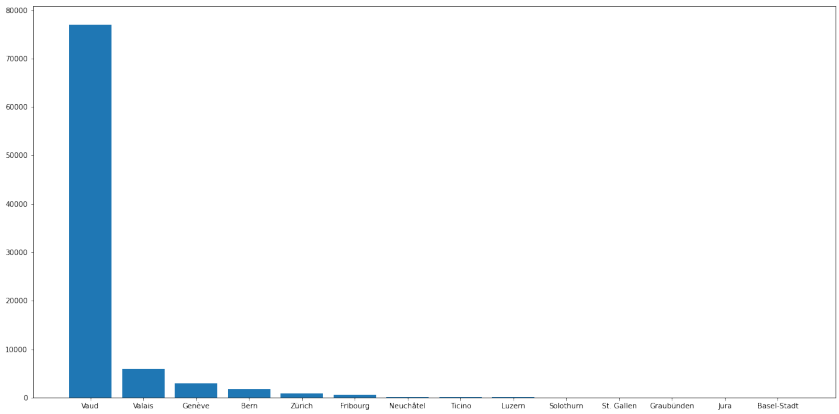


Figure 3.8. Data points distribution over the Switzerland's cantons.

Because the majority of the locations reported in the dataset are located in the Lausanne area, the Switzerland area has been subdivided first accordingly with the different cantons. Second, the Lausanne area has been subdivided into grids of 500m x 500m to provide a finest subdivision of the zone. The choice of this grid size is imposed by the infeasible number of zones that would have been created using a finer grid. This choice it is also compatible with the methodology proposed in[10]. Together, cantons and Lausanne grids, generate a total of 423 different zones with 396 zones dedicated merely to the Lausanne area.

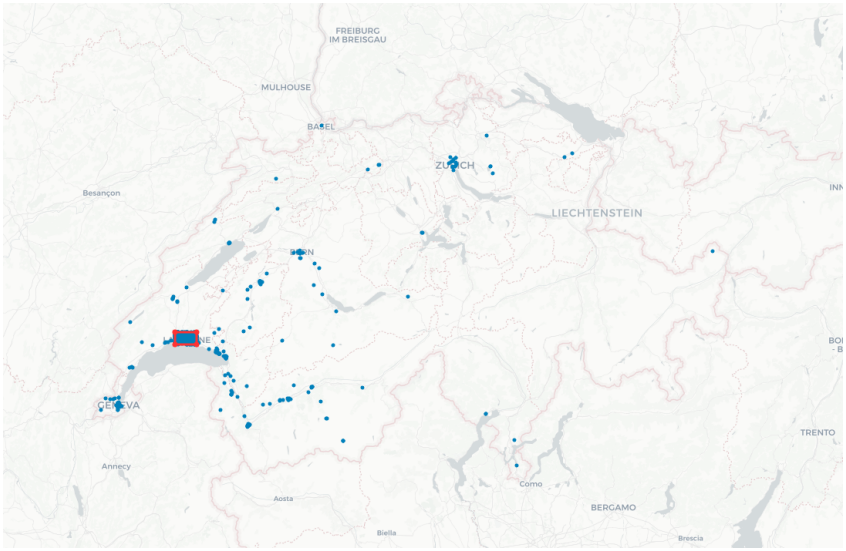


Figure 3.9. Data points distribution over the Switzerland's area.

As can be seen from the histogram in figure 3.8, 86% of the registered data points are located in the canton of Vaud, where Lausanne is located. Over the 86% of the records located in the canton of Vaud, almost the 95% of the data points fall in the Lausanne area.

The major limit of this dataset is it data sparsity. The analysis of the pre-processed records showed that on average, between two consecutive locations of a single user, there are 16 hours.

This statistic however is unbalanced by long periods of time in which some users have not collected data which can have last up to over 5,000 hours.
These *'time skips'* could not be removed or fixed, removing a time skip means removing also all the successive records, while trying to fill the gap would implicated the addition of a high number of artificial records. Further analysis showed how the 35.8% of the records are within 1 hour of distance between each other, the 74% can be found within the 12 hours while the 87.6% can be found within a day (24 hours).

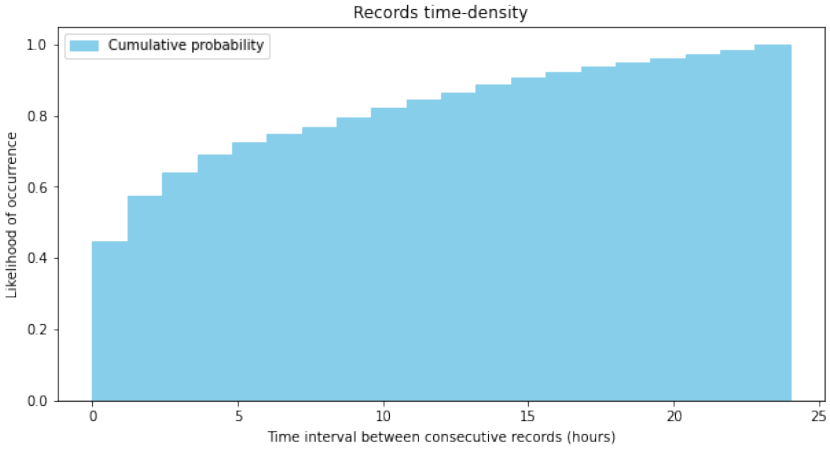


Figure 3.10. Cumulative distribution of the time differences between consecutive data points.

Despite its relatively low density and the limited number of records, the Nokia LDCC dataset provides the required information such as both the users' mobility traces and their demographic data. This allows to overcome the implicit limitations provided by the synthetic mobility traces produced by PERSONA. The final pre-processed dataset is composed by six features illustrated in table 3.13, which provides also the features' description and the range of values covered by each.

Name	Description	Value range
user_id	Unique user id	-
age_group	Age of the user	[1, 8]
working	Working status	[1, 8]
weekday	Day of the week	[1, 7]
month	Month of the year	[1, 12]
zone_id	Id of the zone where the user is located	[0, 422]

Table 3.13. Final pre-processed Nokia LDCC dataset schema.

Chapter 4

Design and performance analysis of federated learning models for human mobility

4.1 Introduction

The main advantage of the TensorFlow Federated framework is the possibility of easily adapt and use normal Keras models in the federated settings by just wrapping them into a callable function. For this reason, to conduct a research FL simulation, it's good practice to first develop a good centralized deep learning model to be used then with proper federated settings. TensorFlow strongly recommends writing and testing TF logic as simple functions. In this way, the TF logic can be executed and tested under a controlled environment before serializing the computation for TFF that, as anticipated in subsection 2.6.2, does not rely on Python. Once the logic has been serialized, it loses some debugging information while eventual feedbacks relative to error and problems could be more opaque and difficult to understand.

Following this direction and the promising work proposed in DeepMove[3] and PMF[10], a Keras model based on recurrent neural networks has been developed to test out the capabilities of this approach and compare the performance improvements provided by the usage of GRU layers respect to simple feed-forward ANN. This chapter covers the choice of recurrent neural networks over feed-forward neural networks; The influence of demographic data in the human mobility prediction tasks and the results obtained by applying federated learning to human mobility models. Furthermore, the federated learning paradigm has been tested through the analysis of different factors such as the communication cost; The sparse availability of clients; Its possible role in fine-tuning pre-trained models and lastly, the effects of differential privacy on the model utility.

4.1.1 Metrics and methodology

The work presented in this chapter is entirely based on Python and TensorFlow's related frameworks such as TensorFlow Federated, Keras and TensorFlow Privacy. All the experiments have been executed on hosted runtimes provided by Google Colaboratory in both the free and Pro ver-

sions using hardware acceleration when supported. In the following chapter, the terms "*users*", and "*clients*" are both used to refer to individuals participating with their data to the training, validation or evaluation processes for both centralized and federated settings. The experiments results are usually described and represented using two metrics, the *sparse categorical accuracy* and the *sparse categorical cross-entropy loss*. These two metrics will be often defined simply such as *Accuracy* and *Loss* to improve the readability and are defined as follow. The sparse categorical accuracy metric measures how often, or better to say, the percentage of records, for which the predictions matches with integer labels provided as Ground-Truths. To enable the optimization of the models, the sparse categorical cross-entropy has been selected as the loss function. Its definition follows the function defined by the categorical cross-entropy:

$$CCE(y, \hat{y}) = -\sum_{c=1}^{C} y_c log(\hat{y}_c) \tag{4.1}$$

The prefix "*sparse*" of the two metrics simply represent that the categorical targets are not encoded or embedded but just represented as integers.

4.2 Recurrent Neural Networks and Feed-Forward ANNs

To answer the question of which class of neural networks provides better performance with the next-place prediction task, two models have been developed. The feed-forward model is built upon a basic architecture with multiple inputs, an embedding layer, one dense hidden layers and an output layer with the Softmax activation function applied to normalize the output probabilities. This experiment makes use of the NYC Taxi Dataset introduced in section 3.2.
Because the dataset is structured in terms of taxi trips with a pickup and a related dropoff location, it does not need an expensive pre-processing to be used with this architecture. Each trip is also described by some temporal features such as the pickup and dropoff hours, the day of the week and whether or not that day is a weekend. The task of this model is simple, given the pickup location and its temporal information, predict the dropoff location of that trip, or in other words, forecasting the next place visited by the taxi (which represents the user).

Feature engineering is required to provide proper features representation for the model. While the weekend feature is binary, the pickup location is instead categorical and require so to be encoded or embedded. Due to the elevate number of taxi zones, a simple one-hot encoding would be inefficient. A one-hot encoded vector is sparse and with a total of 263 different taxi zones, more than the 99% of elements of the encoded vector would be zero. For this reason, the pickup location feature has been instead embedded. The feature embedding provides an efficient and dense representation of the categorical feature using a dense vector of float values whose length is a hyperparameter of the model. Those values are trainable parameters of the model and are learned and tuned during the training such as the weights. The embedding of the pickup locations provides so a better representation of the input feature but require also more data to be trained.

Additional features such as the pickup hour and day of the week are not purely categorical. Different values, such as two subsequent days of the week, are related to each other by an order relation. Because Monday comes after Sunday, but at the same time it comes before, those features are not categorical but *cyclical*. This kind of features required a specific representation[77] which involves representing the cyclical features as (x,y) coordinates on a circle. The pickup

hour and day of the week are so represented by two new features each, the *sine* and the *cosine* of a unit circle.

$$hour_{sin} = sin(hour * (2 * \frac{\Pi}{24}))$$
$$hour_{cos} = cos(hour * (2 * \frac{\Pi}{24}))$$

Figure 4.1. Sine and Cosine components computation example.

This representation makes the features easier to be understood and exploited by the neural network improving the performance[77]. Table 4.6 shows the input features of the models, while the ground truth for each example is provided by the *dropoff_location* of the trip.

Name	Type	Value range
pickup_location	Categorical	[0, 262]
pickup_hour_sin	Numerical	[-1, 1]
pickup_hour_cos	Numerical	[-1, 1]
pickup_weekday_sin	Numerical	[-1, 1]
pickup_weekday_cos	Numerical	[-1, 1]
pickup_weekend	Numerical	[0, 1]

Table 4.1. Feed-forward model input features.

The hyperparameters used for the training are defined in table 4.2.

Batch Size	Dense Units	Embedding Size	Optimizer	Learning Rate
128	128	256	Adam	0.001

Table 4.2. Hyperparameters used for training the feed-forward ANN.

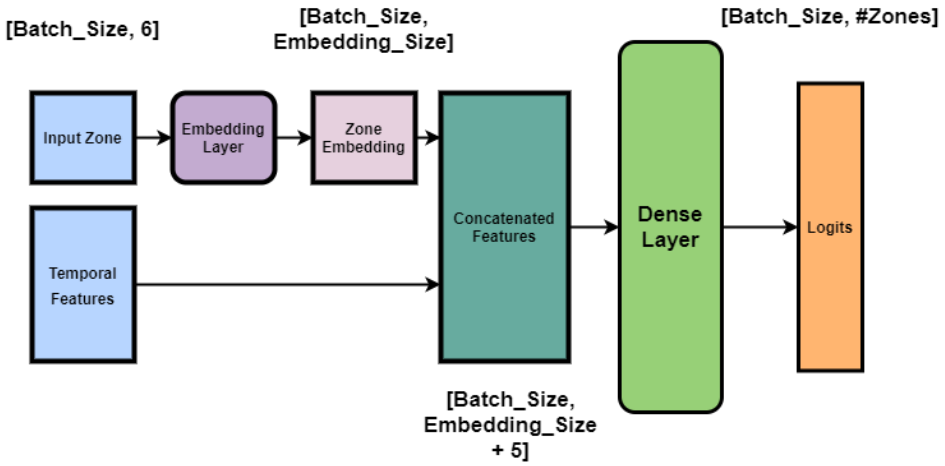


Figure 4.2. An overview of the used feed-forward neural network.

The recurrent model based on recurrent neural networks uses instead a hidden layer of Gated Recurrent Units (GRU)[78], A type of RNN related to the classic LSTM. Both GRU and LSTM utilize different ways of gating information to prevent the vanishing gradient problem, but GRU is relatively new, with performance on par with LSTM, while computationally more efficient having a simpler internal structure. While to feed a feed-forward ANN the dataset has not required any particular pre-processing, RNNs require to have sequences such as inputs and targets, and so the dataset has been split into sequences of N locations.

To better understand the process, table 4.3 provides an example.

id	pickup_location	dropoff_location	pickup_time	dropoff_time
1	A	B	1	2
2	B	C	2	3
3	D	E	4	5

Table 4.3. Simplified example of three records.

In order to create the sequences of locations which will be fed as input into the model, the locations present in the table should be 'flattened' into a simple list of unique locations (e.g. [A, B, C, D, E] in this example). Due to the sparsity of mobility traces, the dropoff location of a record could not coincide with the pickup one of the next trip in many cases. This creates problems of duplicated locations, considering the two location columns of the table, the location identified by *B* would appear twice, and so an occurrence of it must be removed. To solve this issue, during the generation of the sequences, each pickup location has been checked for being, or not, a duplicate of the previous dropoff location and in case, removed. Table 4.4, following the given example, illustrates the structure of the processed dataframe before being split into sequences.

id	location	time
1	A	1
2	B	2
2	C	3
3	D	4
3	E	5

Table 4.4. Expected output from the flattening process of the example proposed in table 4.3.

Each batch fed into the model is a three-dimensional tensor of shape [*Batch_size*, *sequence_length*, *#features*]. The input features are the same used by the feed-forward model just presented. To preserve the internal state across different batches, the GRU layer has been made stateful. This requires all the batches to have the same size, and so incomplete batches (batches with a dimension smaller than the fixed batch size) have been discarded.

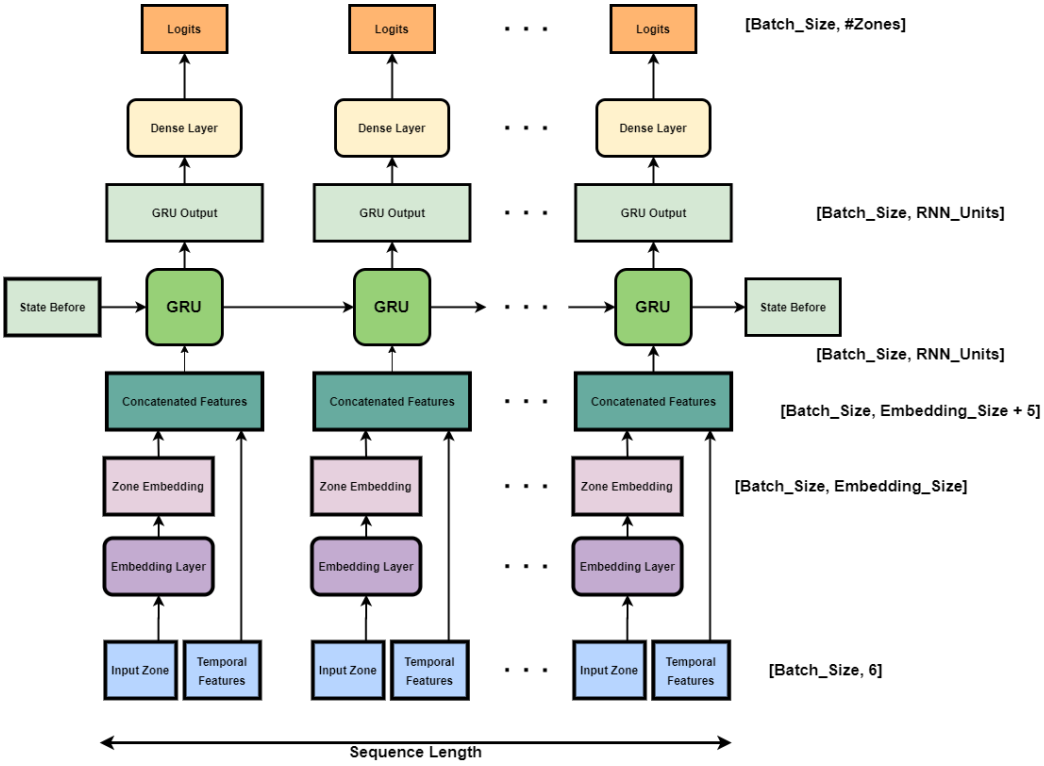


Figure 4.3. An overview of the used recurrent neural network.

The hyperparameters used for the training are defined in table 4.5, while table 4.6 lists the input features if the model.

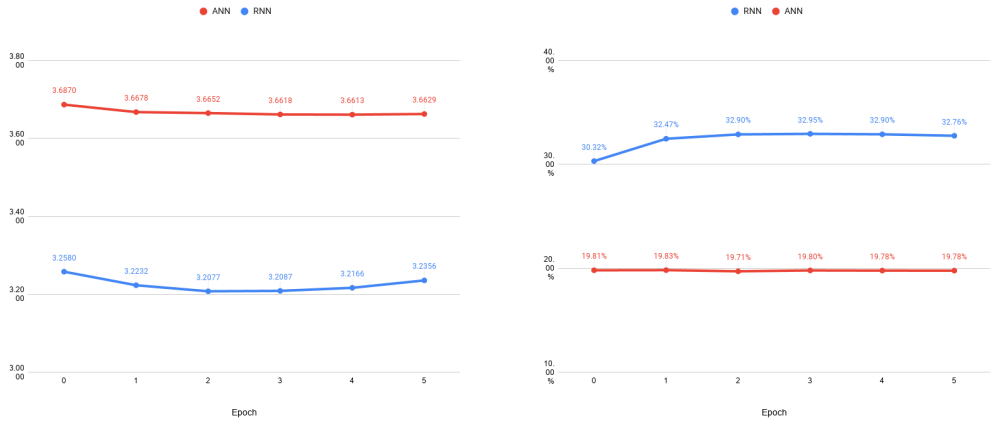
Batch Size	RNN Units	Embedding Size	Optimizer	Learning Rate	Sequence Length
16	256	256	Adam	0.002	16

Table 4.5. Hyperparameters used for training the RNN.

Name	Type	Value range
location	Categorical	[0, 263]
hour_sin	Numerical	[-1, 1]
hour_cos	Numerical	[-1, 1]
week_day_sin	Numerical	[-1, 1]
week_day_cos	Numerical	[-1, 1]
weekend	Numerical	[0, 1]

Table 4.6. Recurrent model input features.

To fairly train the models and compare their performance, the experiment has been designed in the same way for both the models. The data of each taxi has been split into *train*, *validation* and *test sets*. All the sets of each taxi have been then joined together to create the complete sets with data of all the taxis. To better assess the model performance with an unbiased set of data, records of five extra taxis not part of the training set have been used to generate a second *extra-test set*. The two models use so the same data but pre-processed in different ways due to the input sequences required by the recurrent model. Hyperparameters such as the learning rate; The network architecture and the embedding sizes have been selected through many tests to fine-tune the models and find the optimal combination of hyperparameters. Although these are the optimal hyperparameters found, better combinations can exist but this is beside the scope of this work. The training process of the models have been monitored by the usage of TensorFlow callbacks such as early-stopping and logging with TensorBoard.



(a) Sparse Categorical Cross Entropy Loss (b) Sparse Categorical Accuracy

Figure 4.4. Comparison between the performance of the two models on the validation set during training.

The training performance showed in figure 4.4 give an overview of the loss and accuracy achieved by the two models on the validation set. During the five epochs of training, the model based on recurrent neural networks outperformed the simple feed-forward model with an average improvement over 10% in accuracy, achieving also a lower loss value.

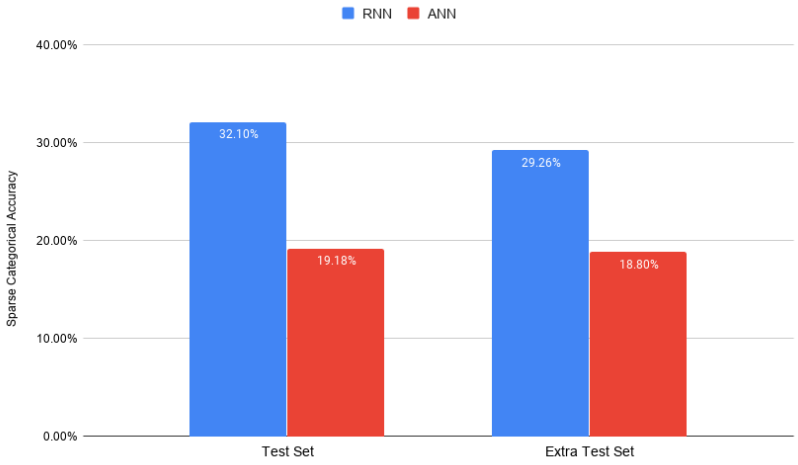


Figure 4.5. Evaluation results of the two models on the two test sets.

Figure 4.5 shows the evaluation results highlighting the improvement provided by the usage of recurrent neural networks over simple feed-forward ANN-based models for the next-place prediction task. The ability of the RNNs to keep trace of past locations and transitions enables the model to reach a sparse categorical accuracy that in our tests was able to exceed the 30% on both the test sets, while the feed-forward model stops around the 19%, confirming the

advantages provided by the usage of recurrent neural networks over feed-forward ANNs.

4.3 The influence of demographic data

In the previous section, in order to train and evaluate the models, all the taxis' data have been used together to feed the model. While taxis have not routines or relevant places such as home and workplaces, real people have them and they move accordingly. Zheng et al. 2012 showed that human mobility is associated with relatively fixed locations, and are characterized by a strong regularity in the long term. An individual's movement pattern is so related to its habits and social identity. This suggests that individuals who share parts of their identity with others can also be related by some mobility behaviors. For example, the mobility patterns of a young student could share more similarities with traces of another student of its city than with an employee of a different country.

It is so possible that predictors trained on a specific group of people whose members share some similarities can achieve better performance when applied to new individuals similar to that group than a population model which should work for everybody. The robustness of this intuition has been discussed by a recent work of Baumann et al.[73] in which the authors concluded that only a little advantage can be obtained by developing different mobility models for different groups of users.

Therefore, to further investigate this hypothesis and test the robustness of a population model built upon data of all the users, the demographic information provided by the Nokia LDCC dataset, and the synthetic traces produced with PERSONA have been used to build groups and population models.

Because has not been possible to work on the LDCC dataset until August, The PERSONA dataset has been created purely to generate strongly characterized groups of users. Individuals of the same group in PERSONA share so many aspects of their routines and relevant places, emphasizing the intra-group low entropy. For this reason, the results provided by the experiments run on this dataset have to be considered as proxies of the results obtained on the real traces of the Nokia LDCC.

To perform the experiments, four different models have been trained and evaluated on the dataset PERSONA. While the first model has been fed with data of all the sixty users in the dataset, the latter three have been trained with data of one demographic group each (*Employee*, *Retired* and *Student*). Each model is based on recurrent neural networks with a very similar architecture to what explained in the previous section despite the usage of one-hot encoding instead of the embedding layer due to the limited number of places which are only 13. The training process, and the dataset splitting into train, validation and test sets, follow the same rules applied in the previous section.

Each user in PERSONA is described by an attribute which specifies its demographic group (or class as it has been defined in the dataset introduction section 3.3), and the records of the three demographic groups have been created accordingly by selecting only the users' records of those groups from the complete dataset. Table 4.7 illustrate the number of records available for each model or group.

Model	Users	Records	Fraction
Population	60	41,760	100%
Employee	26	18,096	43%
Retired	16	11,136	27%
Student	18	12,528	30%

Table 4.7. Fraction of records available for each model or group.

The training process of each model has been automatically interrupted by an early-stopping policy, while the weights which allow each model to achieve the best performance during the training phases have been saved to avoid evaluation on an overfitted model. Table 4.8 lists the model input features, while Table 4.9 illustrate the used hyperparameters.

Name	Type	Value range
place_id	Categorical	[0, 12]
hour_sin	Numerical	[-1, 1]
hour_cos	Numerical	[-1, 1]
weekend	Numerical	[0, 1]

Table 4.8. Models input features.

Batch Size	GRU Units	Optimizer	Learning Rate	Sequence Length
32	128	Adam	0.001	32

Table 4.9. Hyperparameters used for the experiments.

Figure 4.6. Comparison between the performance of the three groups models and the population model on their respective validation sets during training.

All the models have been validated and trained on their related fractions of the dataset. The Employee models has been so trained, validated and finally evaluated on records of users of the Employee class. While the population model has used all the records available. Figure 4.6 shows the differences in terms of sparse categorical accuracy on their respective validation sets during the training phase. The Employee group model seems to provide the largest performance improvement with an accuracy of up to 3 percentage points over the population model. Despite this, the Student model shows performance very similar to the population model and the Retired model seems instead to suffer even more its deficit of data with performance up to 3 percentage points lower than the population model considered as baseline. The average performance of the three group models is instead very similar, if not identical, to what achieved by the baseline model.

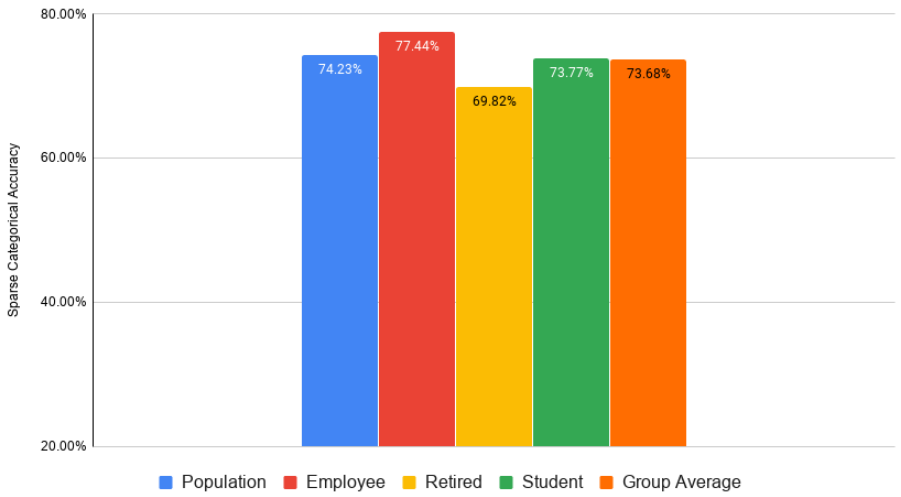


Figure 4.7. Evaluation results of the four models on their respective test sets, together with the average between the three groups.

While figure 4.7 illustrates evaluation's performance very similar to those observed with the validation sets, more interesting results are provided by figure 4.8, that shows the evaluation result of the population model over the test sets of the different groups. Each group model is so tested against its group's test set and the population model is evaluated on the same sets to provide a direct comparison on the same data.

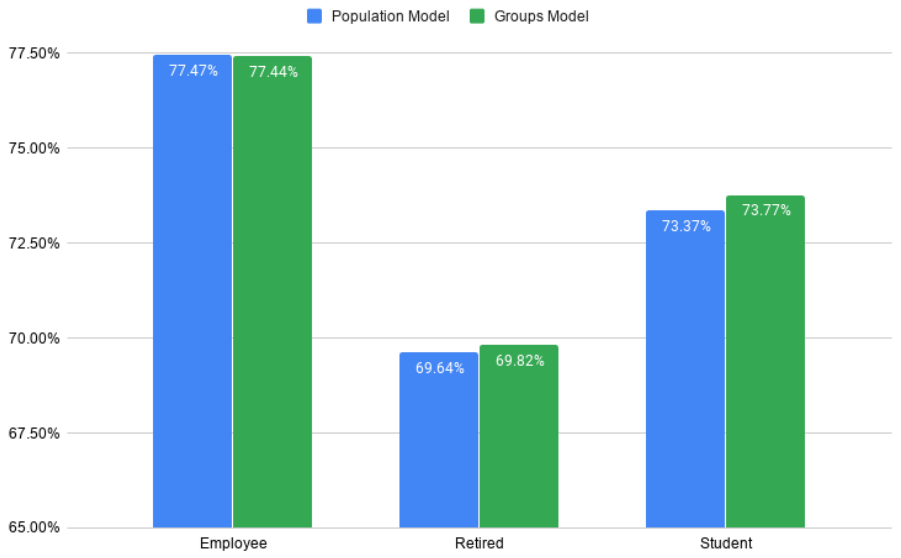


Figure 4.8. Comparison of evaluation performance of the population model with the group models over their test sets (PERSONA dataset).

Despite the previous disparity in the performance of the models, with the Employee group model outperforming the population base model, when evaluated on the same set of data, the population model performs almost identically to all the group models. With a difference in sparse categorical accuracy up only to the 0.5%, the population model demonstrates its robustness over all the three different demographic groups, even surpassing the Employee model's performance which outperformed it of the 3% in the previous tests.
To further investigate these counter-intuitive results, and stress them on real data, successive experiments have been conducted on the real mobility traces provided by the Nokia LDCC dataset. The definition of the demographic groups for the Nokia LDCC dataset have followed two different attributes of the dataset. The *working* attribute describes the main occupation of the user, which has been classified into eight different categories listed in Table 4.10. User's demographic data also provide the *age_group* feature. The range of possible ages has been split into eight buckets of similar sizes.
Table 4.10 better illustrates the different values and their frequencies in the dataset. These two features have been combined to create groups of users whose members are in the same age group and share the same occupation to increase the similarities between users in the same group. It is important to note that groups composed of less than four users have not been considered in these experiments. A group with just a few users becomes more similar to an individual model than a group one able to generalize over a wide number of users. Table 4.10 also lists the *working-age_group* combinations.

Label	Description	Records number	Frequency
W_A	working full time	48,276	60.9%
W_B	working part time	4,758	6.0%
W_C	not currently working	1775	2.2%
W_D	studying full time	21,033	26.5%
W_E	studying part time	0	0%
W_F	housewife	2,621	3.3%
W_G	retired	804	1.0%
W_H	other	0	0%
AG_A	less than 16 years old	0	0%
AG_B	between 16 and 21 years old	5,823	7.3%
AG_C	between 22 and 27 years old	29,862	37.3%
AG_D	between 28 and 33 years old	25,188	31.4%
AG_E	between 33 and 38 years old	11,817	14.7%
AG_F	between 39 and 44 years old	3,623	4.2%
AG_G	between 45 and 50 years old	3,796	4.7%
AG_H	more than 50 years old	0	0%
$AG_D W_A$	between 28 and 33 years old and working full time	19,636	29.16%
$AG_C W_D$	between 22 and 27 years old and studying full time	13,208	19.61%
$AG_C W_A$	between 22 and 27 years old and working full time	12,788	18.99%
$AG_E W_A$	between 33 and 38 years old and working full time	9,876	14.66%
$AG_B W_D$	between 16 and 21 years old and studying full time	5,397	8.01%
$AG_D W_D$	between 28 and 33 years old and studying full time	2,316	3.44%
$AG_D W_B$	between 28 and 33 years old and working part time	2,197	3.26%
$AG_F W_A$	between 33 and 38 years old and working full time	1,932	2.86%

Table 4.10. Demographic data in the LDCC dataset.

Many demographic categories do not have a proper amount of data to be used in these experiments, these groups have not been considered.

The architecture of the models is still based on the best-performing recurrent neural network found in section 4.2. The sequence of input locations, or zones, are first embedded into a dense representation by the Embedding layer. The concatenation of the embedding and the numerical features is then fed into the stateful GRU layer directly connected to the output dense layer which counts 423 neurons with a Softmax activation function. Table 4.11 lists the model input features and Table 4.12 illustrates the used hyperparameters.

Name	Type	Value range
zone_id	Categorical	[0, 422]
hour_sin	Numerical	[-1, 1]
hour_cos	Numerical	[-1, 1]
week_day_sin	Numerical	[-1, 1]
week_day_cos	Numerical	[-1, 1]
weekend	Numerical	[0, 1]

Table 4.11. Models input features.

Batch Size	RNN Units	Embedding Size	Optimizer	Learning Rate	Sequence Length
8	256	256	Adam	0.001	8

Table 4.12. Hyperparameters used for the experiments.

The training process of each model has been automatically interrupted by an early-stopping policy, while the weights which allowed each model to achieve the best performance during the training phases have been saved to avoid evaluating an overfitted model.

As did with PERSONA, after the training phase each group model has been tested against its group's test set and the population model is evaluated on the same sets of the group models to directly compare the achieved sparse categorical accuracy. Figure 4.9 shows that the results are very similar to what has been obtained with the PERSONA dataset. Over the eight different groups, only in three cases the population model achieves the worst performance with a maximum difference up to the 10% on the AG_FW_A (users between 33 and 38 years old and working full time) group. While in 5 cases it achieves better or equivalent performance.

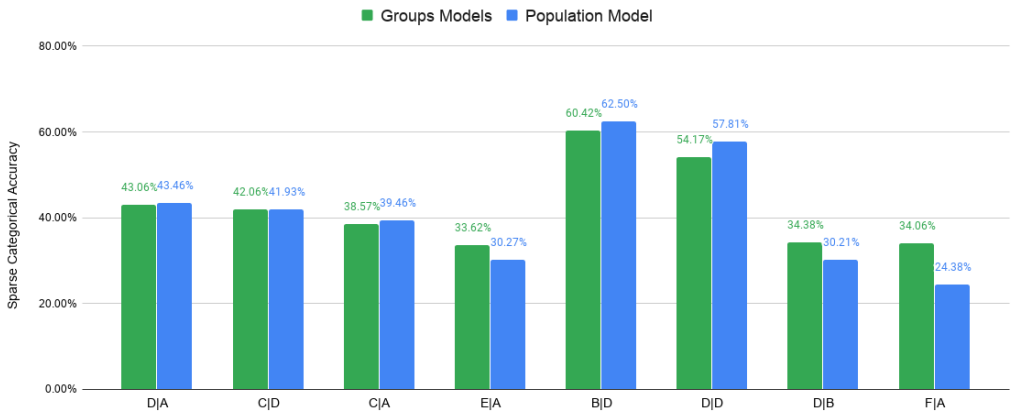


Figure 4.9. Comparison of evaluation performance of the population model with the group models over their test sets (Nokia LDCC dataset).

Figure 4.10 illustrates the average difference in accuracy between the population and the

group models for each group's number of unique users. The trend line suggests that with the increase of the number of users in the groups, the performance of the population and the group models tends to converge. Due to the limited size of the available dataset, it has not been possible to further investigate this hypothesis.

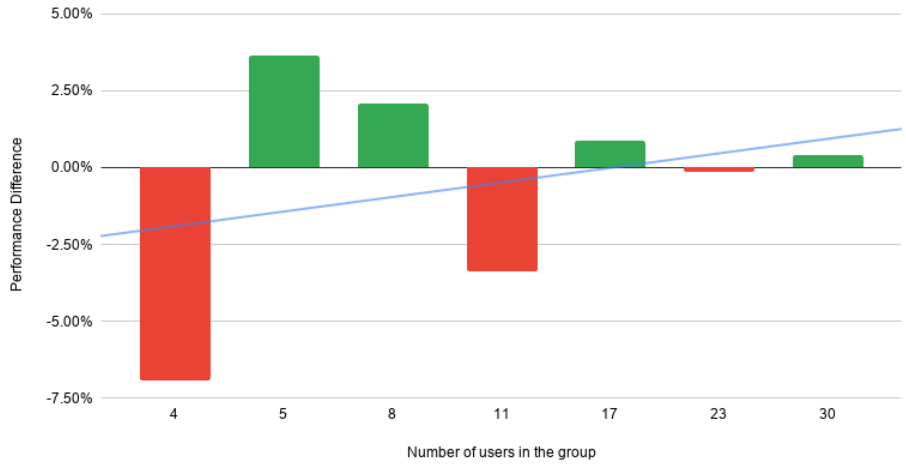


Figure 4.10. Average difference in sparse categorical accuracy between the population and the group models for each group's dataset size computed in terms of number of users.

The results of the experiments conducted on two different datasets show that the developed population model can achieve comparable performance to those provided by building joint models for different groups of users with common demographic characteristics for at least the 62.5% of the groups. Confirming the outcomes achieved by Baumann et al. in 2018. Furthermore, the similarities between the proxy results obtained on the synthetic traces of PERSONA, and those given by the successive experiments on the LDCC dataset, confirm the validity and the effectiveness of the proposed synthetic traces generation method, at least for this scope. In conclusion, it seems so more effective to build local individual models starting from a federated model trained upon the whole population, than generate and handle many different federated models for all the possible demographic groups.

4.4 Human mobility models via federated learning

To federate a Keras model, it needs to be wrapped in an instance of the Model interface provided by TensorFlow Federated, which exposes methods to monitor and control training and federated processes. The federated training process is subdivided into communication rounds, and each round is composed of two different levels of model optimization. Because data live in the edge devices, at each round the federated model is first trained locally at the edge level on the clients' devices which have been selected for the round. Then, once all the local model updates have been computed on the devices, they are sent, together with the training metrics, to the server which creates the global model update by averaging them using the federated averaging algorithm (introduced in section 2.5.1). To simulate a federated environment, the datasets must

be federated as well. Data of individual users have to be processed individually, as if they were truly distributed over different placements. Because data are unbalanced, not independent, or identically distributed (not-IID), some users may have to be excluded due to the lack of available records. Each user's dataset should include no concept of a device or user identities. For this reason, all the users' datasets are inserted into a common list without any reference to the data owner. There's no notion of devices or clients identities in the federated computations or the underlying runtime. All the federated computations are expressed as operations on all the clients, without any reference to the computation participants.

The NYC Taxi dataset (Section 3.2) and the Nokia LDCC (Section 3.4) have so been federated by considering the 100 taxis with more data for the first, and the 70 users with more records for the latter. The median number of records per taxi in the first dataset is 2,424, while the LDCC stops to just 941. Albeit with a centralized approach, model's hyperparameters such as the input sequence length and the batch size can be selected by looking at the performance achieved with different values; In a federated environment, it can happen that not all the users have a sufficient amount of records to fill a complete batch. For this reason, the choice of those parameters must also take into account the amount of data owned by the users and not only the model's performance.

The first experiment aims to compare the performance achieved by the application of the privacy-preserving federated learning paradigm to the baseline provided by the centralized model. For each dataset, two models have been built, a centralized and a federated model. Both the centralized and the federated models share the same architecture, which is identical to what has been illustrated in the previous section 4.3. Despite all the models share the same architecture, the hyperparameters are indeed different and are listed in tables 4.13 and 4.14. These models' settings have been also used for all the successive experiments.

Batch Size	GRU units	Optimizer	Learning Rate Client	Learning Rate Server	Sequence Length
16	256	Adam	0.002	0.06	16

Table 4.13. Hyperparameter used with the NYC taxi Dataset. In the centralized mode, the learning rate used is the client's one.

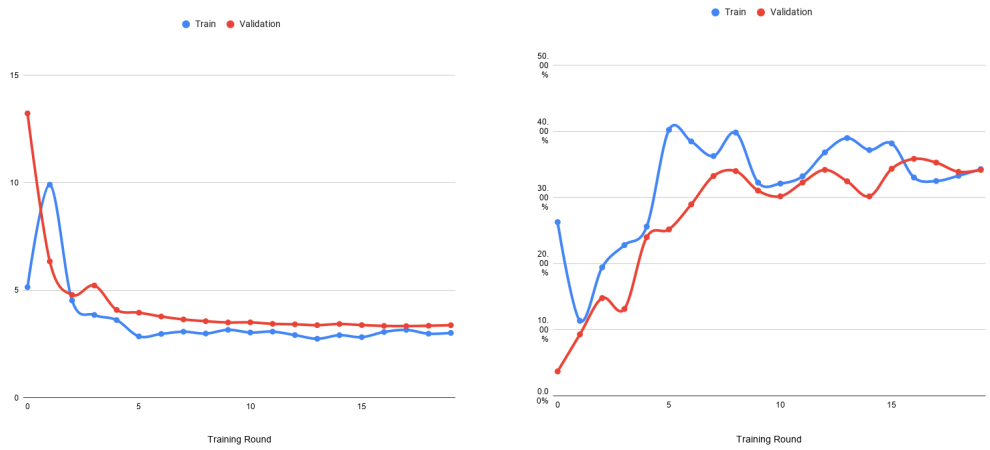
Batch Size	GRU units	Optimizer	Learning Rate Client	Learning Rate Server	Sequence Length
8	256	Adam	0.001	0.06	8

Table 4.14. Hyperparameter used with the Nokia LDCC Dataset. In the centralized mode, the learning rate used is the client's one.

As before, the hyperparameters used for the training are the best found through many experiments with different combinations. Training, validation and test sets have been created in the same way for both the centralized and the federated models, they include the same data points and can be used to fairly compare the models' performance. In addition to those, another unbiased extra-test set with data of clients not used in the training phase is built for both the NYC taxi dataset and the Nokia LDCC.

4.4.1 Federated learning of Nokia LDCC

The training phase of all the models has been monitored through TensorBoard and regulated by early-stopping on the validation loss. Figure 4.11 shows the plots of train and validation losses and sparse categorical accuracy for the federated model during the training phase.

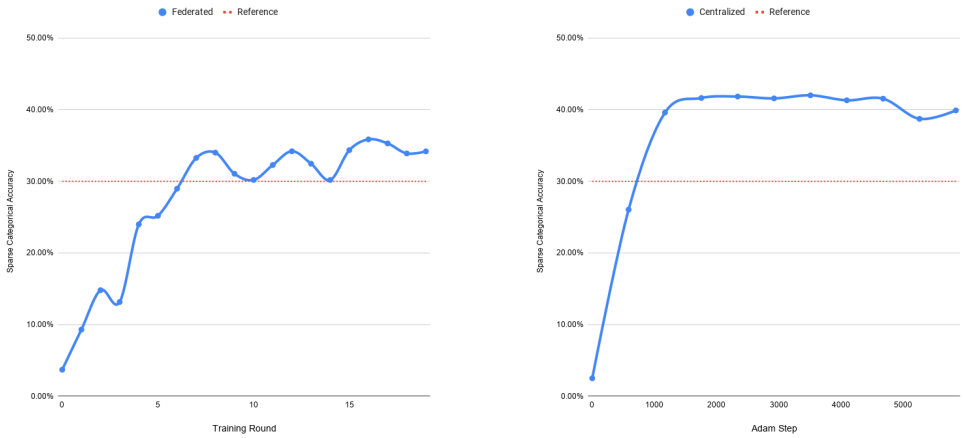


(a) Sparse categorical cross entropy loss as function of the training rounds.

(b) Sparse categorical accuracy as function of the training rounds.

Figure 4.11. Loss and Sparse Categorical Accuracy of the federated model during training.

The early-stopping policy has stopped the training after 19 communication rounds to prevent the upcoming overfitting. Despite the fine-tuning, the federated model has never shown in any test to achieved an accuracy higher than what the centralized model is able to achieve on any set of data available. Figure 4.12 shows a comparison between the validation sparse categorical accuracy reached by both the models during the training process.



(a) Sparse categorical accuracy on validation set as function of training round during federated training.

(b) Sparse categorical accuracy on validation set as function of training Adam step during federated training.

Figure 4.12

While the centralized model reaches an accuracy level of 40% after just two epochs, to then start converging around the 42%, the federated model never achieves the 40% before starting overfitting on the training sets. The federated training process appears also less regular than what achieved by the server-trained model which does not present fluctuations if not when it starts overfitting. This can be caused by both federated averaging, which weights the clients' updates considering their local dataset size, and the federated evaluation process which validates the updated models after training directly on the clients and then averages the results. Figure 4.13 shows the evaluation results of both the models on the two different test sets.

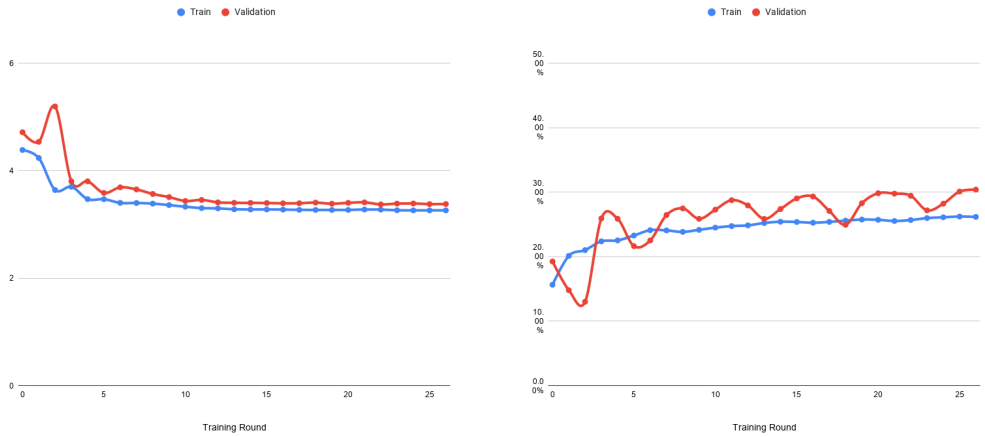
Figure 4.13. Evaluation results of the two models on the two test sets.

The average delta in accuracy between the federated and the centralized model remains

stable around 5%. It worth notice that in a simulation environment federated computations such as the training process do not benefit of the computational power of different devices. Instead, such processes are disadvantaged by the additional computations required to train different models, average the updates, update the global model and so on. This, together with the lack of optimization of the TensorFlow Federated framework which is not able to take full advantage of multiple GPUs, has impacted the number of possible trials for the optimization of hyperparameters. Although the proposed performance are based on fine-tuned parameters, these could be considered as sub-optimal in the hyperparameters' domain space.

4.4.2 Federated learning of NYC Taxis

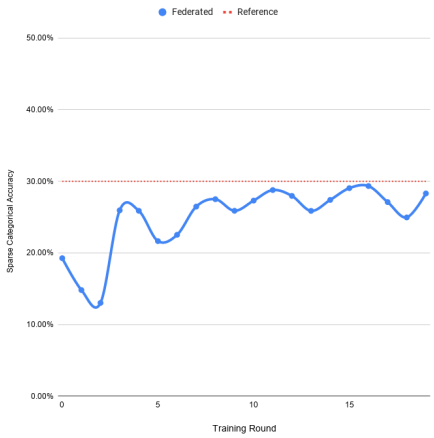
The results obtained on the second dataset follow the same line with the federated model failing to achieve the same level of performance provided by the centralized. While the centralized model reaches easily the 30% of accuracy after the first training epoch, the federated model converges just below that level, for then start overfitting, at the twenty-sixth round. Also in this case, the federated training process appears less regular than what achieved by the server-trained with noticeable fluctuations in the validation accuracy.



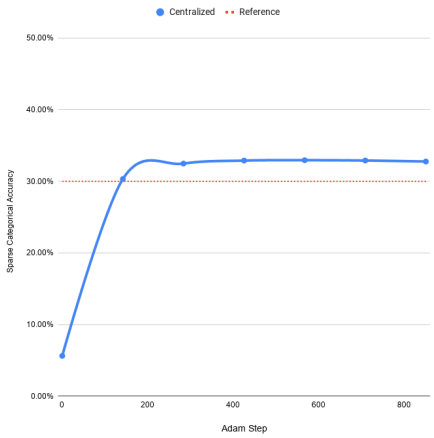
(a) Sparse categorical cross entropy loss as function of training rounds.

(b) Sparse categorical accuracy as function of training rounds.

Figure 4.14. Loss and Sparse categorical accuracy of the federated model during training.



(a) Sparse categorical accuracy on validation set as function of training round during federated training.



(b) Sparse categorical accuracy on validation set as function of training Adam step during federated training.

The average delta in accuracy between the federated and the centralized model is in this case less relevant than what has been observed on the first dataset, with a value around the 2%. Figure 4.16 shows the evaluation results of both the models on the two different test sets.

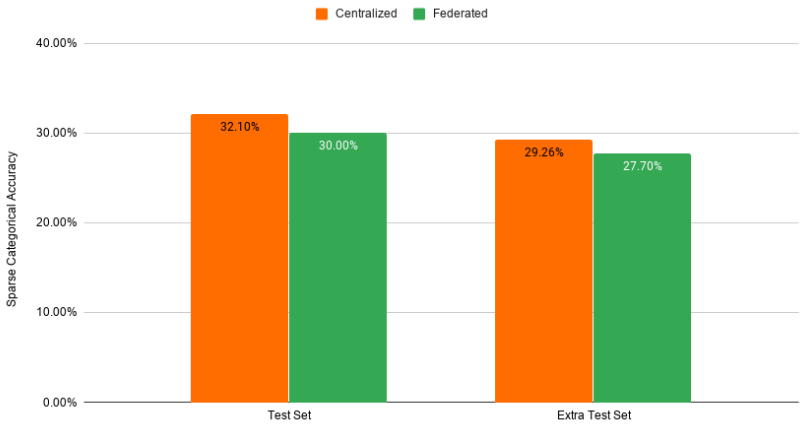


Figure 4.16. Evaluation results of the two models on the two test sets

Despite the loss in prediction performance, federated learning achieved promising results. The advantages are multiple, from the high level of privacy protection provided to the users and not having to collect and manage huge amount of sensitive data, to the computational cost which is exponentially reduced by the availability of multiple devices on which the model building and evaluation can be performed. Surely, the gap in performance between the centralized and the federated model can be reduced through additional optimization, but in none of the tests performed federated learning has provided better performance than standard deep learning methods.

4.4.3 Complexity and communication overhead

One of the main challenges federated learning has to face is that every client participating in the training round have to upload its full model (or a full model's update) back to the server. While in a common centralized machine learning scenario the computational cost dominates, even with the recent introduction of GPU's computing, in a federated context the communication cost dominate, with all the common mobile devices problems related to the connectivity. Though in a simulated environment this problem does not subsist, this can represent a bottleneck in practical scenarios. The uplink collection speed is typically much slower than the downlink necessary to download the model. Additionally, privacy measures like the application of cryptographic protocols to the updates further increase the amount of data transmitted.

To verify if a naive version of federated learning without any kind of model compression provides any advantage in terms of communication cost for the clients, the NYC Taxi dataset has been used to perform further analysis.

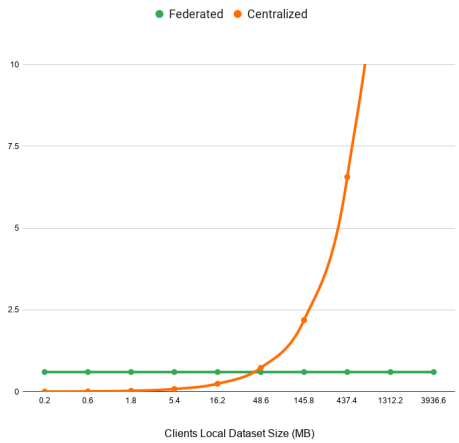
The Keras model considered for this experiment is the same used for the previous test on the NYC taxi dataset and counts 534,024 model's parameters. Table 4.15 lists the shape and the number of parameters per each layer of the model.

Layer	Shape	Number of parameters
Numerical Inputs	[16, 16, 5]	0
Categorical Input	[16, 16, 1]	0
Sequence Embedding	[16, 16, 256]	67,584
Concatenate	[16, 16, 261]	0
GRU	[16, 16, 256]	398,592
Dense Output	[16, 16, 264]	67,848
Softmax	[16, 16, 264]	0
Total		534,024

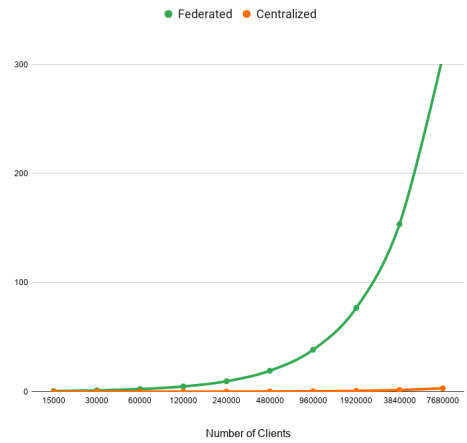
Table 4.15. Summary of the model's parameters.

Knowing that each parameter is represented as a *tf.Float32* which is composed of 32 bits, the weight in memory of the model's parameters is around 2.0 MBytes. Supposing that the full model's update is sent, and ignoring any possible communication protocol headers, at each round all the clients selected for training receive and then send 2.0 MB each back to the server. Considering the 100 taxis in the dataset with more records in January 2013, the average number of records is 2,432; Where each record is composed by the six input features of the model plus the taxi medallion. The average weight in memory for this amount of data stored as a CSV file is 275 KB. The number of yellow cabs in NYC is equal to 13,587 taxis, supposing a stable number of records at each month, one year of data would weight 3.3 MBytes for each taxi. The size of a centralized dataset created by collecting the local yearly data from all the taxis would be around 43 GBytes. Federated learning would require instead to send and then collect 26 GBytes of model parameters at each training round. Considering 26 as the number of training round necessary to achieve the model with the smaller validation loss as observed in our experiments, the total amount of data uploaded by each taxi in the federated environment would be 52 MB; Much more than with a centralized approach. While strategies for improving communication efficiency exist such as structured and sketched updates[80], it is clear that whether or not

federated learning provides any advantage in the communication cost strongly depends on the relation between the size of the clients' local datasets and the model's size. A distributed dataset composed of images, or even videos, would more likely be used with federated settings due to the large files' size. On the other hand, if the number of clients becomes huge, collecting and managing huge amount of data in a unique location can also be difficult. Terabytes of data flowing together to a server can generate network congestions and faults. Federated learning can so avoid the big data management problems despite a possible higher communication cost.



(a) Total amount of data uploaded expressed in Terabytes in function of the average clients' datasets considering a fixed model size of 2 MB and 15,000 clients



(b) Total amount of data uploaded expressed in Terabytes in function of the number of clients considering a fixed model size of 2 MB and 20 communication rounds

Figure 4.17. Comparison between the estimated total amount of data uploaded to train the centralized and the federated models.

4.5 The influence of clients availability with unbalanced and non-IID data

In a real scenario, clients are eligible for the training round only when under a stable and unmetered internet connection, connected to a power source and have a local dataset of sufficient size. Because in a simulated environment all the clients are available for every round, their sparse availability needs to be simulated. To do this, at each training round a fraction of C clients are selected randomly to participate. The selected clients receive the global model and perform two epochs of training on their local data and then send the updates back to the server for aggregation. The model is evaluated after each round on the validation set built from all the users to provide a comparable performance evaluation across all the models at any round. Table 4.16 lists the values of C used for the experiment expressed in percentage of the 70 users part of the Nokia LDCC training set.

Model Number	C	Number of users
1	20%	14
2	40%	28
3	60%	42
4	80%	56
5	100%	70

Table 4.16. Fraction of users at each training round per model.

The training phase of all the models has been monitored through TensorBoard and regulated by early-stopping on the validation loss. Figures 4.18 and 4.19 show the plots of validation loss and accuracy at each round for every value of C tested.

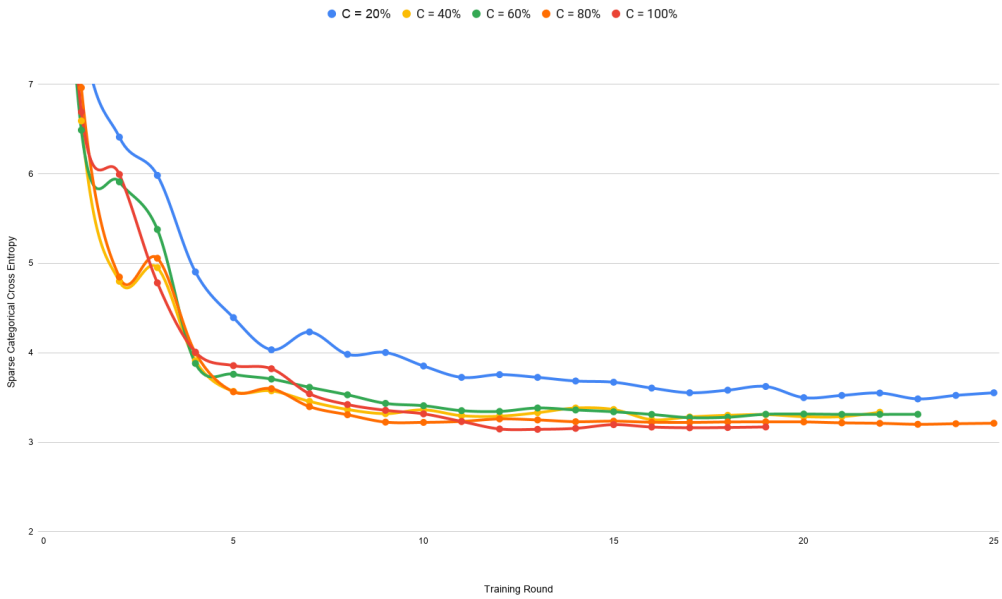


Figure 4.18. Sparse categorical cross entropy loss on validation set as function of training round during federated training.

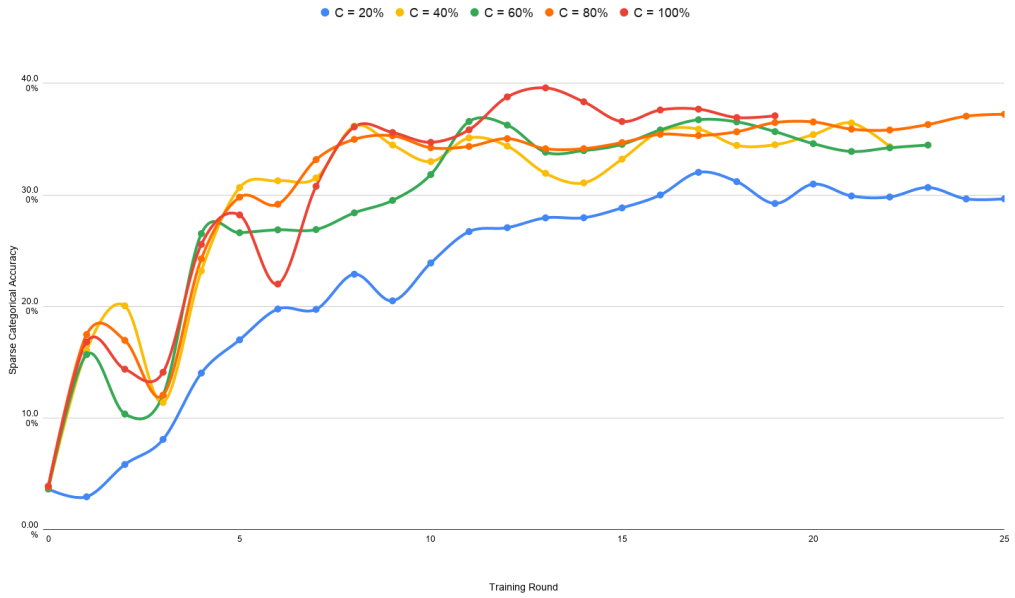


Figure 4.19. Sparse categorical accuracy on validation set as function of training round during federated training.

Training results show that higher values of C tend to provide faster convergence and higher performance by achieving lower loss values and higher accuracy. Since clients are selected at random in this simulation, it can happen that at some round a model is trained on clients which do not well represent the entire population. This can delay the convergence of a model, as in the case of the model with $C = 80\%$ which has shown to require more training round than the others despite the higher number of clients used per round. Figure 4.20 shows instead the evaluation results of the models on the same two test sets.

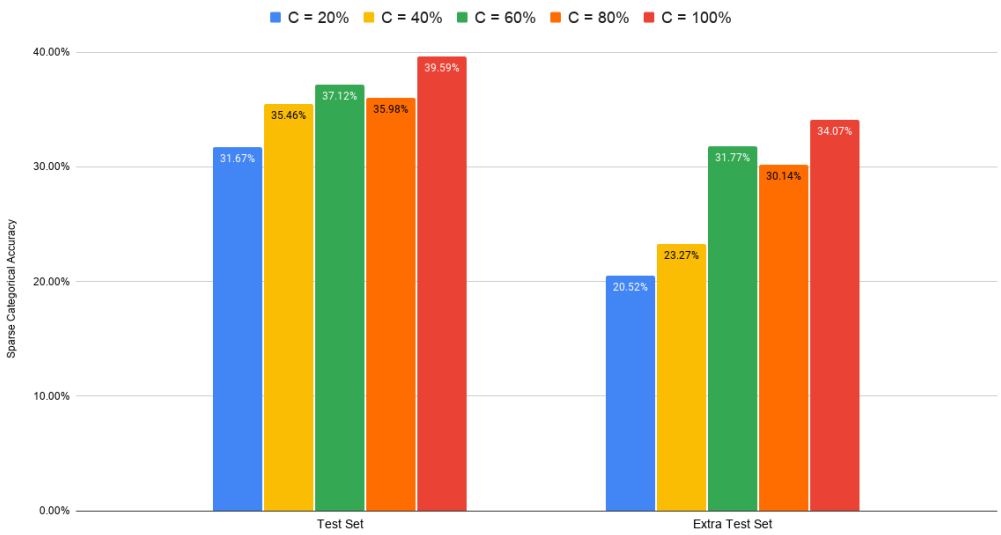


Figure 4.20. Evaluation results of the models on the two test sets.

Results suggest that higher values of C, and so more clients participating at each round, provide both a better end performance and also a better generalization of the model on previously unseen data such as those included into the extra-test set. While the gap in accuracy between the models on the first test set is not particularly relevant, the evaluation on the extra-test set suggests that to obtain a good generalization of the model, at least the 60% of the clients should be eligible at each round.

4.6 The advantages of centralized proxy data

The existence and availability of pre-trained models have made popular transfer-learning techniques making training models from scratch rarely necessary. Transfer-learning makes it possible to take advantage of the knowledge gained by a trained model on a specific domain by allowing rapid progress or improved performance on a different but similar domain. It is so possible to start from a pre-trained model and fine-tuning it using federated learning on decentralized data for a similar or identical task. Let's assume for example that the central server already has mobility data of some clients which gently donated them for the purpose. These data can be used to pre-train a model on the server that then will be refined through federated learning, protecting the privacy of all the other clients.

The assumption is so that not starting from scratch, a federated model would require less time to converge and provide better performance. The experiments have been conducted as follow. The Keras model is first server-trained on data of a subset of C users of the 70 users of the Nokia LDCC dataset for E epochs. Then, the model parameters are saved and loaded into the global federated model which is trained on the remaining $FC = 70 - C$ users who didn't participate to the pre-training session. The federated training continues for R rounds regulated by an early-stopping policy. Table 4.17 illustrate the training settings of each model.

Pre-train number of users (C)	Number of federated clients (FC)	Pre-train epochs (E)	Federated Rounds (R)
0	70	9	14
10	60	9	17
20	50	9	16
35	35	9	11
50	20	9	13

Table 4.17. Experiments settings of the models.

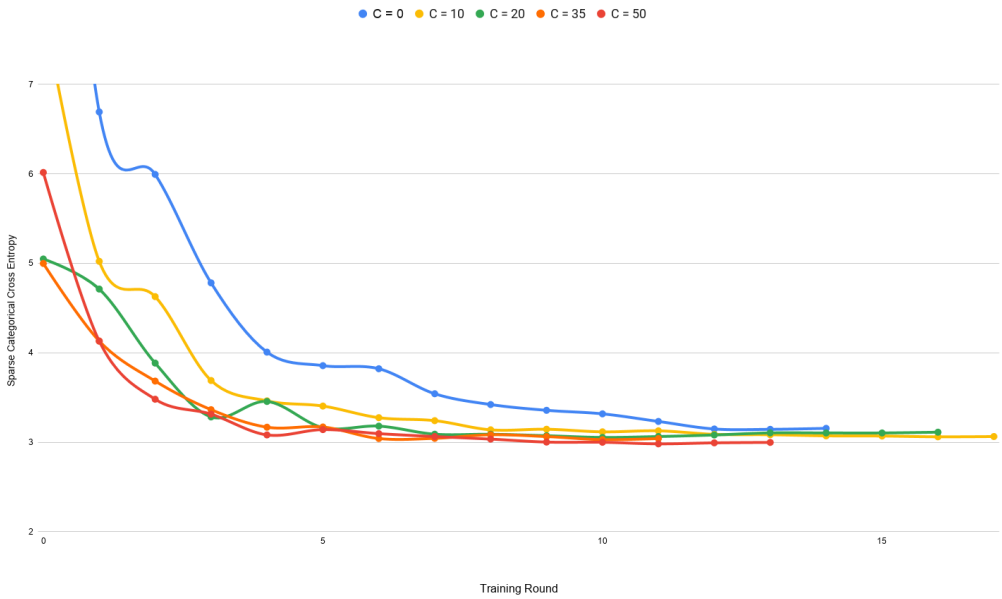


Figure 4.21. Sparse categorical cross entropy loss on validation set as function of training round during federated training.

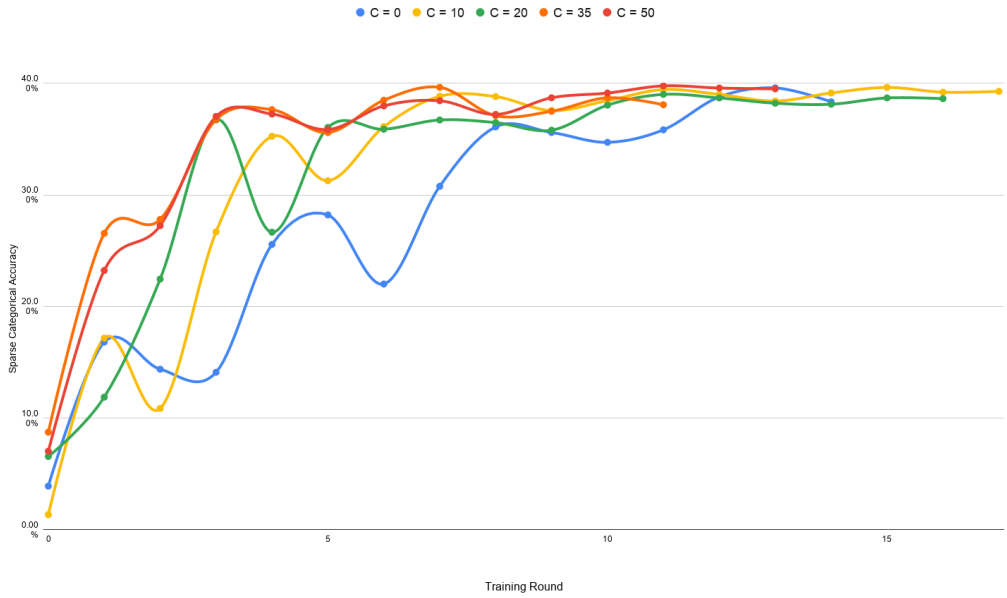


Figure 4.22. Sparse categorical accuracy on validation set as function of training round during federated training.

Figures 4.21 and 4.22 show the speedup provided by pre-training a federated model. While the full-federated model, that represents the baseline, achieves a validation loss close to the value of 3.0 in 12 training rounds, the pre-trained models take almost half of that time to do the same. Models pre-trained with a higher number of clients tend to achieve better performance faster, although increasing C from 35 to 50 seems not to provide any relevant advantage. Thi is probably due to the smaller number of clients left for the subsequent federated training.

Pre-train number of users (C)	Test set accuracy	Extra test set accuracy
0	39.59%	34.07%
10	39.67%	34.81%
20	39.45%	34.67%
35	39.86%	35.51%
50	38.91%	34.70%

Table 4.18. Evaluation results.

Evaluation results on both the test sets in table 4.18 further confirm the advantages provided by the pre-train, with pre-trained models improving the results achieved by the baseline. In conclusion, the results of the experiments show that having pre-trained models can speed up the federated learning process by both reducing the number of training rounds required to converge (also reducing the communication cost) and providing better overall performance. Transfer-learning techniques can also benefit from the federated learning paradigm by adapting the model's knowledge to new domains while preserving the privacy of the users.

4.7 The effects of differential privacy on federated learning performance

The TensorFlow Privacy Python library, introduced in section 2.7, provides methods and implementations of TensorFlow optimizers, such as Adam and SDG, to train TensorFlow models with differential privacy. It worth notices that in federated settings, following the notations presented by McMahan et al. in[81], a *record* is considered as a single client's model update, and a sample as a set of participating clients (or users) for the round. While in a centralized context a record is often described as the average gradient on one microbatch and a sample as a minibatch (a set of microbatches).

TensorFlow Privacy can be applied to TensorFlow Federated by implementing custom DP algorithms and applying them to the clients' updates before average them with federated averaging. As it has already been introduced in section 2.7, DP works by both clipping the updates norms to ensure that gradients have a maximum Euclidean norm, and multiplying the clipped updates by random noise. While in a non-differentially private context the gradient of the loss, with respect to the model parameters, is multiplied by the learning rate and applied to update the parameters, with differential privacy the gradient is first clipped and then random noise is added. The clipped and noised gradients are then multiplied by the learning rate and applied to update the parameters.

Differentially private optimizers require two additional hyperparameters, the *L2 norm bound* which control the clipping and *noise multiplier* to control the amount of noise added to the gradients.

To measure the privacy guarantee achieved many parameters are taken into account. These are the noise multiplier; The probability of a client being selected in a training round; The size of the dataset and the number of steps the global optimizer takes to update the global state.

Privacy analysis measures estimate how likely the presence, or the absence, of a single client's data in the training changes the probability of learning a set of parameters.

The level of differential privacy guaranteed can be measured by two parameters, *epsilon* (ϵ) and *delta* (Δ). Delta limits the probability of the privacy guarantee not holding, and it is usually set to the inverse of the population size. Epsilon instead measures the strength of the privacy provided and gives an upper bound on how much the inclusion, or exclusion, of a single record can increase the probability of a specific output. A smaller value of epsilon corresponds to better practical privacy.

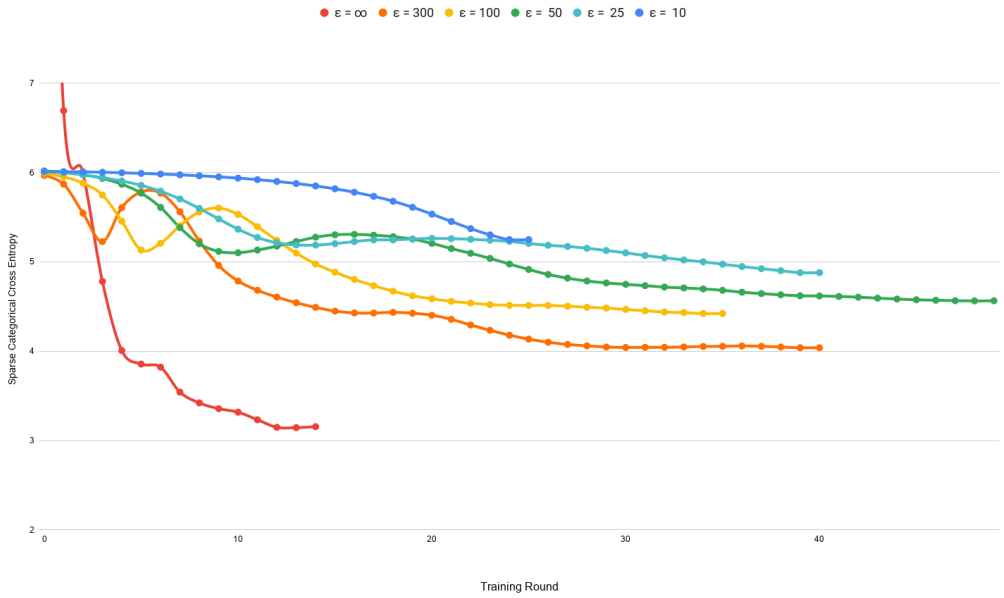
The experiments have followed the procedure described by McMahan et al.[82] that requires to first find the largest value of noise multiplier that allows achieving a good utility during training on a small set of clients per round, and then, while increasing the number of clients, proportionally increase also the noise multiplier. The clipping threshold, which influences neither delta nor epsilon, has been set empirically to 1.0 after several tests and as suggested in[83].

The *privacy/utility trade-off* asks to choose between a higher privacy guaranteed to the users and better performing models. The DP parameters just presented act as knobs to trade-off between a smaller epsilon and a higher accuracy of the model predictions. The scope of the conducted experiments is to further explore the relationship between the utility of the federated model and the privacy risks to which a user is exposed. Table 4.19 lists the differential privacy parameters while illustrating various utility/privacy trade-offs achieved by each model.

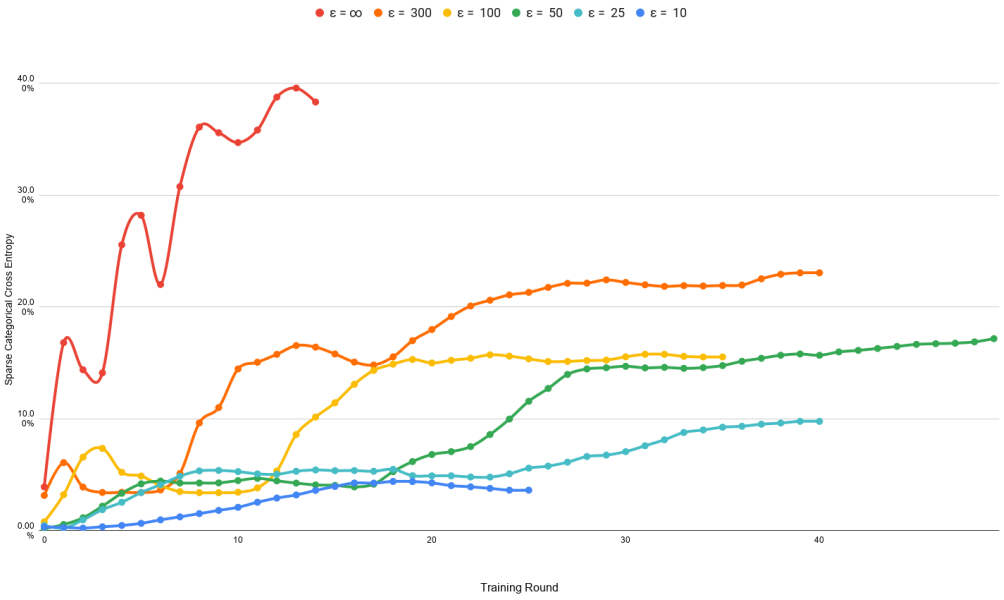
Delta	Clipping Threshold	Number of Clients	Noise Multiplier	Rounds	Epsilon	Accuracy
-	-	70	-	14	∞	36.83%
1e-2	1.0	70	0.3	40	300	28.65%
1e-2	1.0	60	0.5	35	100	21.55%
1e-2	1.0	70	1.0	60	50	20.8%
1e-2	1.0	60	1.3	40	25	14.95%
1e-2	1.0	70	1.7	25	10	3.85%

Table 4.19. Differential privacy parameters used for the six models. Accuracy refers to the average sparse categorical accuracy achieved by each model on the test set and the extra-test set.

Where epsilon equal ∞ represents the baseline federated model where no differential privacy is applied. The strength of the privacy provided (epsilon) has been measures by the integrated methods provided by TensorFlow Privacy which are based on the Rényi divergence to measure distance between distributions[84]. All the models have been trained upon the same training set such as for the validation and the evaluation phases. The only difference between the models is represented by the differential privacy parameters adapted to provide the desired level of epsilon, and the server learning rate which decreases with the increase of the noise level added.



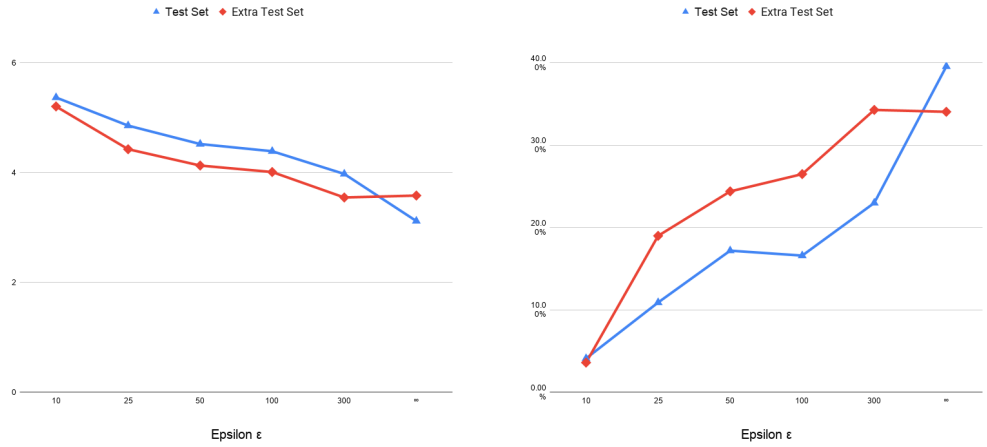
(a) Sparse categorical cross entropy loss on validation set as function of training round during federated training.



(b) Sparse categorical accuracy on validation set as function of training round during federated training.

Figure 4.23. Loss and Accuracy comparison on the validation set of federated models with different levels of privacy protection (ϵ).

Figure 4.23 gives an overview of the training phase for each of the models. It is immediately clear that noisy and clipped updates hardly influence the models' performance. Not only the federated model without any differential privacy measure converges in less than the double number of communication rounds required by all the models implementing differential privacy, but also achieved a much higher level of accuracy. When ϵ reduces below 25, the level of accuracy achieved on the test set does not even reach 10% and so the level of utility provided drops drastically. While results on the validation set seem to suggest that even with a small level of privacy protection the performance drops significantly, more interesting results are provided by the evaluation phase on the test set and the extra-test set, with the latter built from data of unseen users as did in all the previous experiments.



(a) Sparse Categorical Cross Entropy Loss on the two test sets as function of ϵ.

(b) Sparse Categorical Accuracy on the two test sets as function of ϵ.

Figure 4.24. Loss and Accuracy comparison on the two test sets of federated models with different levels of privacy protection (ϵ).

While the performance achieved on the extra-test set have always been lower than on the test set in all the previous experiments[4.16][4.20][4.5], models implementing differential privacy at any level show instead the opposite. Figure 4.24 show the evaluation results for all the models in terms of loss and accuracy achieved as function of ϵ. Only the baseline model, with no differential privacy, exhibits a lower loss and a higher accuracy on the test set than on the extra one.

Data	$\epsilon = \infty$		$\epsilon = 300$		$\epsilon = 100$		$\epsilon = 50$		$\epsilon = 25$		$\epsilon = 10$	
	Acc.	Improv.	Acc.	Improv.	Acc.	Improv.	Acc.	Improv.	Acc.	Improv.	Acc.	Improv.
Test Set	39.59%	0	23.00%	-41.89%	16.60%	-58.06%	17.20%	-56.55%	10.90%	-72.46%	4.10%	-89.64%
Extra Test Set	34.07%	0	34.30%	+0.67%	26.50%	-22.21%	24.40%	-28.38%	19.00%	-44.23%	3.60%	-89.43%

Table 4.20. Sparse categorical accuracy achieved and improvement in percentage respect the baseline for each value of ϵ.

Table 4.20 shows the accuracy and the provided improvement over the non-DP baseline

achieved for each level of ϵ tested on both the test sets.

The improvement columns confirm the better performance achieved by the models implementing DP on the second test set respect to the former with a utility loss which is more than halved in the majority of the cases. Only with the lower value of epsilon tested the utility loss achieved on the two test sets becomes almost identical. Since the main purpose of differential privacy is to make the model independent from private information of individuals, while withholding sensitive data by clipping and adding noise to the model updates, DP also reduces the knowledge that can be acquired by the model specifically about the users participating of the training. This, together to the limited number of possible unique users that can be selected for each round due to the limited size of the training set, has led to a limited learning capability on the users involved in the training phase. On the other hand, the generalization ability of the model shows fewer drawbacks and models adapt properly to new previously unseen data such as the one included into the extra-test set. In conclusion, the bad overall performance achieved by enabling differential privacy at almost all the ϵ could be mainly due to the limited number of users and data available in the used dataset. On a larger scale, the utility loss and the privacy protection provided can probably be mitigated and improved thanks to a larger amount of data and unique users.

Chapter 5

Conclusions, Limitations and Future Work

This thesis investigated the application of the novel privacy-preserving federated learning paradigm to the field of human mobility modelling.

Using three different datasets, we first investigated the problem of modelling human mobility from a non-federated perspective. This allowed us to understand the advantages provided by recurrent neural networks over classic feed-forward netural networks and the implications of building different mobility models for different demographic groups. Then, with solid foundations provided by the findings of this first part of the work, we proposed a federated human mobility model, which achieves encouraging performance while protecting the privacy of the users by allowing their sensitive data not to leave the local devices.

The federated paradigm has been then examined under different perspectives, including: The communication cost, which dominates in a decentralized environment; The influence of the sparse availability of clients; The advantages of pre-training with centralized proxy data and the application of possible transfer-learning techniques. Lastly, we investigated the effects of differential privacy on model prediction performance. The application of differential privacy has resulted in performance below our expectations. We suppose that this privacy-preserving measure requires larger datasets to better preserve the performance of the model while providing additional privacy guarantees. Having a larger number of users with more data would probably make it possible to achieve higher levels of privacy protection (lower levels of epsilon ϵ) while providing higher utility.

Federated learning has revealed promising performance in our experiments besides the little loss in utility with respect to classic, centralized learning techniques. The advantages are multiple: The high level of privacy protection provided to the users; The possibility to avoid having to collect and manage huge amount of sensitive data; The reduction of computational cost, which is reduced by the availability of multiple devices on which the model building and evaluation can be performed. However, advantages come at a cost. The reduced computational cost is replaced by the high communication cost caused by the sharing of the model parameters between the central server and the clients. Additionally, federated optimization systems should also face and solve various major practical problems such as the sparse and different availability of the clients, also considering their environments, or technical problems with the updates, such as the failure of single or multiple updates.

In conclusion, we believe that federated learning will be soon widely applied in many fields, also beyond human mobility modelling. It has the characteristics to change how machine learning models are built and while at the moment no solid and complete frameworks are available, they will be soon. TensorFlow Federated is not even at its 1.0 release and it is still missing a complete integration of differential privacy, and secure aggregation is not available yet. Furthermore, federated averaging is only the second model aggregation strategy developed, new ones will be developed soon and higher performance, reliability and privacy guarantees will be definitely achieved. The federated learning paradigm will create breeding ground for applications which rely on very sensitive data such as in health care and medicine, or whenever transferring data from the edge to the cloud is too expensive such as for multimedia files such as photos and videos.

5.1 Contributions

This thesis provides the following contributions:

- **Description of the core concepts** around human mobility modelling and federated learning,necessary to understand the research context in which this thesis is matured.
- **Presentation of the three datasets**, alongside a description of the followed pre-processing phases to clean the datasets and make them usable with neural networks.
- **Development of PERSONA**, a flexible generator of synthetic human mobility dataset.
- **Comparison and investigation of different neural network architectures** such as feed-forward neural networks and recurrent neural networks applied to the next-place prediction task.
- Study and testing of the **implications of building different mobility models for different demographic groups** using both synthetic and real mobility datasets.
- **Development of a federated mobility model** able to predict the next place visited by a user given the history of its previous visited places.
- **Study of the characteristics and limitations of federated learning** in different scenarios such as the partial availability of clients at each training round; The speed-up in performance provided by pre-training on proxy data and an analysis of the communication cost.
- Study of **the effects of additional privacy-preserving measures such as differential privacy** when applied to the federated learning training process.

5.2 Limitations

This section discusses the shortcoming of this thesis. Each limitation is described by its cause, how it effects the proposed work and possible solutions to overcome it.

- One of the main limitations encountered in the development of this thesis are **the long training times required to build federated models**. In a simulated environment, only one CPU (or GPU) is available and the computations are not parallelized. At each training round, a number of Keras models equal to the number of selected clients is trained and validated. This is a very a computationally expensive task, which can take hours depending on the number of training epochs performed on each clients; The number of clients involved at each training round and the total number of training rounds. To overcome this limitation, while not available yet, high-performance simulations will be enabled soon by TensorFlow Federated framework allowing researches to run simulations on multiple GPUs or multiple machines.

- **The quality and the availability of the datasets** have been the second major issue of this work. Large scale and comprehensive mobility dataset are rare. The NYC Taxi dataset provides dense and numerous traces, but it does not reflect real users' movements with characteristics such as habits and relevant places. It also does not provide any demographic data that was required for our experiments. While the Taxi dataset was publicly available, the Nokia LDCC Dataset required to be authorized to have access to the data. The bureaucracy combined with the delayed time caused by the global pandemic, have delayed our access to this dataset, limiting its possible usages for this thesis. Moreover, the quality and the quantity of the data provided by this dataset are inadequate for the needs of this work. Mobility traces are often sparse in time and the availability and the precision of the users' positions were below our expectations. Wider and more comprehensive human mobility datasets are required to provide more accurate and reliable results.

- While **PERSONA** has been useful to compensate for the absence of real traces, its **simplicity makes it also too artificial and limited**. A wider set of classes and locations, together with higher intra-class entropy could make it more useful for different tasks.

- Due to the late availability, the **data cleaning and exploration processes on the Nokia LDCC Dataset have been less accurate than on the NYC Taxi Dataset**. This has led to the presence of some mobility traces that show repeatedly the same location because the users have not moved for a discrete amount of time. Additional integrity and consistency checks would be necessary to provide more reliable results.

- Another limitation is caused by the **restricted selection of metrics used**. While the sparse categorical accuracy is a valuable metric for the task, an additional metric would have provided a wider view over the achieved results. Additionally, mainly due to the first limitation just reported, results do not present the standard deviation and the values reported are the average of only two runs. To overcome these limitations, a Top-K sparse categorical accuracy can be used to provide useful insights and each run of each experiment would have to be repeated multiple times in order to provide the mean and the related standard deviation. Additional insights can be given instead by considering the confusion matrices and additional metrics such as precision, recall and F1 score.

- Lastly, the application of **differential privacy has been limited by the sizes of the available datasets and the high computational cost** required to train a differentially private model which involves high number of rounds. The availability of high-performance simulations and wider datasets could enable advanced experiments and provide better results.

5.3 Future Work

We believe that our work opens up several opportunities for future research, which are briefly described as follows:

- **Smart area subdivision.** Usually, areas such as towns and cities are characterized by relevant and strategical places such as train stations, airports or universities. These places are more or less relevant depending on the context taken into account. A possible development of the proposed work includes an adaptive and smart subdivision of the area of interest. Depending on the application, the area subject of the analysis can be subdivided into zones considering the semantic meaning of different sub-areas instead of using a simple grid as done with the area of Lausanne3.4. While predicting whether a user will go or not to the intersection at the end of the street does not usually provide significant advantages, knowing that she is probably going to a bus or train station, a supermarket or to a residential area does.

- **Secure aggregation and differential privacy.** To increment the level of privacy protection provided to the users, additional protective measures such as the secure aggregation of the model updates and differential privacy would be applied. While this work has covered already the second measure, the level of privacy protection achieved was not sufficient, mainly due to the lack of data and the absence of high-preforming simulations. With the future implementation of secure aggregation in the TensorFlow Federated framework, a larger mobility datasets and the ability of perform simulation on multiple GPUs, higher levels of privacy protection can be achieved while providing a more comprehensive overview of the methodology.

- **Federated learning in practice.** Through the development of a Android application, it would be possible to not only collect real mobility traces on the field generating a more comprehensive mobility dataset, but also to deploy a real application of a federated learning process coordinated by a central server and trained on multiple devices running TensorFlow Lite. While Google is already using FL on the fields with its mobile keyboard, TensorFlow Federated is still in an early stage of its development and it's mainly focuses on simulation environments. Additionally, not every client has to have the same model. A study conducted over different cities could require the central server to manage and share different models depending on the clients' locations or the desired task.

- **Attacks on federated learning.** Federated learning systems can be vulnerable to targeted attacks from malicious clients due to its decentralized nature. Z. Sun et al.[85] have shown how adversaries can back door federated learning by reducing the performance of the model with altered data and model updates poisoning. TensorFlow Federated allows to simulate targeted attacks by implementing attacking algorithms. At the beginning of 2020, Simon Weckert has shown how fake mobility traces can "hack" a big service such as Google Maps[86]. The influence of possible malicious clients contributing to the model with falsified data should be further explored to provide robust federated human mobility models.

5.4 Conclusions

This work represents a first step towards the development of new machine and deep learning methods that minimize privacy risks the users. Our results show promising performance of federated models, while outline the need for further research to obtain satisfactory levels of privacy protection. We believe that with the rapid development of tools and techniques to support the federated learning paradigm, its potential for privacy-preserving human mobility modelling will be unlocked.

Bibliography

[1] Derek Lin. Machine learning, and deep learning in cybersecurity, Feb 2019. URL https://www.exabeam.com/information-security/understanding-the-role-of-artificial-intelligence-machine-learning-and-deep-learning-in-cybersecurity/.

[2] Recurrent neural network, Jul 2020. URL https://en.wikipedia.org/wiki/Recurrent_neural_network.

[3] Jie Feng, Yong Li, Zeyu Yang, Qiang Qiu, and Depeng Jin. Predicting human mobility with semantic motivation via multi-task attentional recurrent networks. *IEEE Transactions on Knowledge and Data Engineering*, page 1–1, 2020. doi: 10.1109/tkde.2020.3006048.

[4] Andrew Hard, Kanishka Rao, Rajiv Mathews, Françoise Beaufays, Sean Augenstein, Hubert Eichner, Chloé Kiddon, and Daniel Ramage. Federated learning for mobile keyboard prediction. *CoRR*, abs/1811.03604, 2018. URL http://arxiv.org/abs/1811.03604.

[5] Keith Bonawitz, Hubert Eichner, Wolfgang Grieskamp, Dzmitry Huba, Alex Ingerman, Vladimir Ivanov, Chloé Kiddon, Jakub Konecný, Stefano Mazzocchi, H. Brendan McMahan, Timon Van Overveldt, David Petrou, Daniel Ramage, and Jason Roselander. Towards federated learning at scale: System design. *CoRR*, abs/1902.01046, 2019. URL http://arxiv.org/abs/1902.01046.

[6] Federated learning: Collaborative machine learning without centralized training data, Apr 2017. URL https://ai.googleblog.com/2017/04/federated-learning-collaborative.html.

[7] Brendan Mcmahan. *Federated Learning, from Research to Practice*. URL https://www.pdl.cmu.edu/SDI/2019/slides/2019-09-05Federated%20Learning.pdf.

[8] H. Brendan McMahan, Eider Moore, Daniel Ramage, Seth Hampson, and Blaise Agüera y Arcas. Communication-efficient learning of deep networks from decentralized data. In *AISTATS*, 2017.

[9] Alexandra Wood, Micah Altman, Aaron Bembenek, Mark Bun, Marco Gaboardi, James Honaker, Kobbi Nissim, David Obrien, Thomas Steinke, Salil Vadhan, and et al. Differential privacy: A primer for a non-technical audience. *SSRN Electronic Journal*, 2018. doi: 10.2139/ssrn.3338027.

[10] Jie Feng, Can Rong, Funing Sun, Diansheng Guo, and Yong Li. Pmf. *Proceedings of the ACM on Interactive, Mobile, Wearable and Ubiquitous Technologies*, 4(1):1–21, 2020. doi: 10.1145/3381006.

[11] Chih-Ling Hsu. Analyze the nyc taxi data. URL https://chih-ling-hsu.github.io/2018/05/14/NYC.

[12] Laurence Goasduff. Gartner says global smartphone sales continued to decline in second quarter of 2019, Aug 2019. URL https://www.gartner.com/en/newsroom/press-releases/2019-08-27-gartner-says-global-smartphone-sales-continued-to-dec.

[13] Wilhelm Kleiminger, Christian Beckel, and Silvia Santini. Household occupancy monitoring using electricity meters. *Proceedings of the 2015 ACM International Joint Conference on Pervasive and Ubiquitous Computing - UbiComp '15*, 2015. doi: 10.1145/2750858.2807538. URL https://dl.acm.org/citation.cfm?id=2807538.

[14] James Scott, A.J. Brush, John Krumm, Brian Meyers, Mike Hazas, Steve Hodges, and Nicolas Villar. Preheat: Controlling home heating using occupancy prediction. In *Proceedings of UbiComp 2011*. ACM, September 2011. URL https://www.microsoft.com/en-us/research/publication/preheat-controlling-home-heating-using-occupancy-prediction/.

[15] Paul Baumann. User context recognition for navigation systems in public transportation. *ResearchGate*, Mar 2012. URL https://www.researchgate.net/publication/254032637_User_context_recognition_for_navigation_systems_in_public_transportation.

[16] John Krumm. Ubiquitous advertising: The killer application for the 21st century. *IEEE Pervasive Computing Magazine*, 100(1):66–73, January 2011. URL https://www.microsoft.com/en-us/research/publication/ubiquitous-advertising-killer-application-21st-century/.

[17] Indi Samarajiva. How korea does contact tracing, Jun 2020. URL https://medium.com/@indica/how-korea-does-contact-tracing-1b2662b5b894.

[18] Ivan Watson and Sophie Jeong. Coronavirus mobile apps are surging in popularity in south korea, Feb 2020. URL https://edition.cnn.com/2020/02/28/tech/korea-coronavirus-tracking-apps/index.html?utm_source=fbCNN&utm_medium=social&utm_term=link&utm_content=2020-02-28T15:28:07&fbclid=IwAR1NQNJU6QTzYbyYbTLtz_onj67zvMayG4hsP_rK72zxhrzzZT__96rnRLY.

[19] Elsevier, 2013. ISBN 9780123943972. doi: 10.1016/c2011-0-07051-5. URL https://www.sciencedirect.com/book/9780123943972/computer-and-information-security-handbook.

[20] C. Song, Z. Qu, N. Blumm, and A.-L. Barabasi. Limits of predictability in human mobility. *Science*, 327(5968):1018–1021, 2010. doi: 10.1126/science.1177170.

[21] Andrea Hess, Karin Anna Hummel, Wilfried N. Gansterer, and Günter Haring. Data-driven human mobility modeling. *ACM Computing Surveys*, 48(3):1–39, 2016. doi: 10.1145/2840722.

[22] Aarti Munjal, Tracy Camp, and William Navidi. Smooth: a simple way to model human mobility. pages 351–360, 10 2011. doi: 10.1145/2068897.2068957.

[23] J. Yoon, M. Liu, and B. Noble. Random waypoint considered harmful. *IEEE INFOCOM 2003. Twenty-second Annual Joint Conference of the IEEE Computer and Communications Societies (IEEE Cat. No.03CH37428)*, 2003. doi: 10.1109/infcom.2003.1208967.

[24] Tristan Henderson and David Kotz. Data citation practices in the crawdad wireless network data archive. *D-Lib Magazine*, 21(1/2), 2015. doi: 10.1045/january2015-henderson.

[25] I. Rhee, M. Shin, S. Hong, K. Lee, and S. Chong. On the levy-walk nature of human mobility. *IEEE INFOCOM 2008 - The 27th Conference on Computer Communications*, 2008. doi: 10.1109/infocom.2008.145.

[26] Juha Laurila, Daniel Gatica-Perez, Imad Aad, Jan Blom, Olivier Bornet, T.-M.-T Do, Olivier Dousse, Julien Eberle, and Markus Miettinen. The mobile data challenge: Big data for mobile computing research. nokia research center. 01 2012.

[27] Tlc trip record data. URL `https://www1.nyc.gov/site/tlc/about/tlc-trip-record-data.page`.

[28] Transport. URL `https://data.gov.sg/group/transport`.

[29] Anastasios Noulas, Salvatore Scellato, Renaud Lambiotte, Massimiliano Pontil, and Cecilia Mascolo. A tale of many cities: Universal patterns in human urban mobility. *PLoS ONE*, 7 (5), 2012. doi: 10.1371/journal.pone.0037027.

[30] Boris Igelnik and Jacek M. Zurada. *Efficiency and scalability methods for computational intellect*. Information Science Reference, 2013.

[31] Wesley Mathew, Ruben Raposo, and Bruno Martins. Predicting future locations with hidden markov models. *Proceedings of the 2012 ACM Conference on Ubiquitous Computing - UbiComp 12*, 2012. doi: 10.1145/2370216.2370421.

[32] Wenhao Huang, Man Li, Weisong Hu, Guojie Song, and Kunqing Xie. Hierarchical destination prediction based on gps history. *2013 10th International Conference on Fuzzy Systems and Knowledge Discovery (FSKD)*, 2013. doi: 10.1109/fskd.2013.6816336.

[33] Jungkeun Yoon, Brian D. Noble, Mingyan Liu, and Minkyong Kim. Building realistic mobility models from coarse-grained traces. *Proceedings of the 4th international conference on Mobile systems, applications and services - MobiSys 2006*, 2006. doi: 10.1145/1134680.1134699.

[34] Abdulrahman Al-Molegi, Antoni Ballesté, and Mohammed Jabreel. Geo-temporal recurrent model for location prediction. 10 2017. doi: 10.3233/978-1-61499-806-8-126.

[35] Xiaoliang Fan, Lei Guo, Ning Han, Yujie Wang, Jia Shi, and Yongna Yuan. A deep learning approach for next location prediction. *2018 IEEE 22nd International Conference on Computer Supported Cooperative Work in Design ((CSCWD))*, 2018. doi: 10.1109/cscwd.2018.8465289.

[36] Reuben Binns, Ulrik Lyngs, Max Van Kleek, Jun Zhao, Timothy Libert, and Nigel Shadbolt. Third party tracking in the mobile ecosystem. *Proceedings of the 10th ACM Conference on Web Science - WebSci '18*, 2018. doi: 10.1145/3201064.3201089. URL `https://arxiv.org/pdf/1804.03603.pdf`.

[37] Vincent Taylor and Ivan Martinovic. *Quantifying Permission-Creep in the Google Play Store*. 2016. URL https://arxiv.org/pdf/1606.01708.pdf.

[38] Cambridge analytica, Jun 2020. URL https://en.wikipedia.org/wiki/Cambridge_Analytica.

[39] 2020. URL http://faculty.uml.edu/sgallagher/Brandeisprivacy.htm.

[40] Katharine Toll. Privacy and Freedom. By Alan F. Westin. New York: Atheneum Press, 1967. 10.00. *Social Work*, 13(4):114–115, 10 1968. ISSN 0037-8046. doi: 10.1093/sw/13.4.114-a. URL https://doi.org/10.1093/sw/13.4.114-a.

[41] Regulation (eu) 2016/679 of the european parliament and of the council. URL https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[42] *The New York Times*, Aug 2006. URL https://www.nytimes.com/2006/08/09/technology/09aol.html.

[43] Yves-Alexandre De Montjoye, César A. Hidalgo, Michel Verleysen, and Vincent D. Blondel. Unique in the crowd: The privacy bounds of human mobility. *Scientific Reports*, 3(1), 2013. doi: 10.1038/srep01376.

[44] 2020. URL https://www.gps.gov/.

[45] K. Lee, S. Hong, S. J. Kim, I. Rhee, and S. Chong. Slaw: A new mobility model for human walks. *IEEE INFOCOM 2009 - The 28th Conference on Computer Communications*, 2009. doi: 10.1109/infcom.2009.5061995.

[46] Andrew I. Schein, Alexandrin Popescul, Lyle H. Ungar, and David M. Pennock. Methods and metrics for cold-start recommendations. *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval - SIGIR 02*, 2002. doi: 10.1145/564376.564421.

[47] James Mcinerney, Jiangchuan Zheng, Alex Rogers, and Nicholas R. Jennings. Modelling heterogeneous location habits in human populations for location prediction under data sparsity. *Proceedings of the 2013 ACM international joint conference on Pervasive and ubiquitous computing - UbiComp 13*, 2013. doi: 10.1145/2493432.2493437.

[48] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Computation*, 9(8):1735–1780, 1997. doi: 10.1162/neco.1997.9.8.1735.

[49] Iksoo Choi, Jinhwan Park, and Wonyong Sung. Character-level language modeling with gated hierarchical recurrent neural networks. *Interspeech 2018*, 2018. doi: 10.21437/interspeech.2018-1727.

[50] URL https://www.google.com/maps.

[51] Keith Bonawitz, Vladimir Ivanov, Ben Kreuter, Antonio Marcedone, H. Brendan Mcmahan, Sarvar Patel, Daniel Ramage, Aaron Segal, and Karn Seth. Practical secure aggregation for privacy-preserving machine learning. *Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security*, 2017. doi: 10.1145/3133956.3133982.

[52] 2018. URL https://federated.withgoogle.com/.

[53] 2020. URL https://www.cs.toronto.edu/~kriz/cifar.html.

[54] 2020. URL https://www.tensorflow.org/.

[55] Tensorflow federated. URL https://www.tensorflow.org/federated.

[56] Custom federated algorithms, part 1: Introduction to the federated core. URL https://www.tensorflow.org/federated/tutorials/custom_federated_algorithms_1?hl=fi.

[57] Tensorflow. Tensorflow privacy. URL https://github.com/tensorflow/privacy.

[58] Tf-Encrypted. Tf-encrypted. URL https://github.com/tf-encrypted/tf-encrypted/tree/master/examples/federated-learning.

[59] H. Brendan McMahan and Galen Andrew. A general approach to adding differential privacy to iterative training procedures. *CoRR*, abs/1812.06210, 2018. URL http://arxiv.org/abs/1812.06210.

[60] Martin 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*, CCS '16, page 308–318, New York, NY, USA, 2016. Association for Computing Machinery. ISBN 9781450341394. doi: 10.1145/2976749.2978318. URL https://doi.org/10.1145/2976749.2978318.

[61] Dingqi Yang, Daqing Zhang, Vincent W. Zheng, and Zhiyong Yu. Modeling user activity preference by leveraging user spatial temporal characteristics in lbsns. *IEEE Transactions on Systems, Man, and Cybernetics: Systems*, 45(1):129–142, 2015. doi: 10.1109/tsmc.2014.2327053.

[62] Chao Zhang, Keyang Zhang, Quan Yuan, Haoruo Peng, Yu Zheng, Tim Hanratty, Shaowen Wang, and Jiawei Han. Regions, periods, activities. *Proceedings of the 26th International Conference on World Wide Web*, 2017. doi: 10.1145/3038912.3052601.

[63] Vijay Pandurangan. On taxis and rainbows, Jun 2014. URL https://tech.vijayp.ca/of-taxis-and-rainbows-f6bc289679a1.

[64] Chris. Foiling nycs taxi trip data, Mar 2014. URL https://chriswhong.com/open-data/foil_nyc_taxi/.

[65] 2020. URL https://pandas.pydata.org/.

[66] 2020. URL https://colab.research.google.com/notebooks/intro.ipynb#recent=true.

[67] 262588213843476. The approximate max/min latitude and longitude for all states and major territories. URL https://gist.github.com/jakebathman/719e8416191ba14bb6e700fc2d5fccc5.

[68] haversine. URL https://scikit-learn.org/stable/modules/generated/sklearn.metrics.pairwise.haversine_distances.html.

[69] Colin Gorrie. Three ways to detect outliers, Mar 2016. URL http://colingorrie.github.io/outlier-detection.html.

[70] The average speed of traffic in midtown manhattan is 4.7 mph. new york thinks it's found a solution, Jan 2018. URL https://www.latimes.com/nation/la-na-new-york-traffic-manhattan-20180124-story.html.

[71] Headsortails. Nyc taxi eda - update: The fast the curious, Feb 2020. URL https://www.kaggle.com/headsortails/nyc-taxi-eda-update-the-fast-the-curious#direct-distance-of-the-trip.

[72] Open Calgary. Nyc taxi zones. URL https://data.cityofnewyork.us/Transportation/NYC-Taxi-Zones/d3c5-ddgc.

[73] Paul Baumann, Christian Koehler, Anind K. Dey, and Silvia Santini. Selecting individual and population models for predicting human mobility. *IEEE Transactions on Mobile Computing*, 17(10):2408–2422, 2018. doi: 10.1109/tmc.2018.2797937.

[74] J. Laurila, D. Gatica-Perez, I. Aad, J. Blom, O. Bornet, Trinh-Minh-Tri Do, O. Dousse, Julien Eberle, and M. Miettinen. The mobile data challenge: Big data for mobile computing research, 2012. URL https://www.semanticscholar.org/paper/The-Mobile-Data-Challenge%3A-Big-Data-for-Mobile-Laurila-Gatica-Perez/8daeecc84fcaf42172cba7ef58e5068fae7bbcbc?p2df.

[75] Wikipedia Contributors. Decimal degrees, Sep 2020. URL https://en.wikipedia.org/wiki/Decimal_degrees.

[76] 2020. URL https://shop.swisstopo.admin.ch/en/products/landscape/boundaries3D.

[77] Oct 2017. URL http://blog.davidkaleko.com/feature-engineering-cyclical-features.html.

[78] Junyoung Chung, Caglar Gulcehre, and Kyunghyun Cho. Empirical evaluation of gated recurrent neural networks on sequence modeling. URL https://arxiv.org/pdf/1412.3555v1.pdf.

[79] Jiangchuan Zheng and Lionel M. Ni. An unsupervised framework for sensing individual and cluster behavior patterns from human mobile data. In *Proceedings of the 2012 ACM Conference on Ubiquitous Computing*, UbiComp '12, page 153–162, New York, NY, USA, 2012. Association for Computing Machinery. ISBN 9781450312240. doi: 10.1145/2370216.2370241. URL https://doi.org/10.1145/2370216.2370241.

[80] Jakub Konečný, H Mcmahan, Felix Yu, Ananda Theertha, Dave Google, and Peter Richtárik. *FEDERATED LEARNING: STRATEGIES FOR IMPROVING COMMUNICATION EFFICIENCY*. URL https://arxiv.org/pdf/1610.05492.pdf.

[81] H Mcmahan, Galen Andrew, Úlfar Erlingsson, Steve Chien, Ilya Mironov, Nicolas Papernot, and Peter Kairouz. *A General Approach to Adding Differential Privacy to Iterative Training Procedures*. 2019. URL https://arxiv.org/pdf/1812.06210.pdf.

[82] H Mcmahan, Daniel Ramage, Kunal Talwar, and Li Zhang. *Published as a conference paper at ICLR 2018 LEARNING DIFFERENTIALLY PRIVATE RECURRENT LANGUAGE MODELS*. URL https://arxiv.org/pdf/1710.06963.pdf.

[83] tensorflow. tensorflow/privacy, Mar 2020. URL https://github.com/tensorflow/privacy/tree/master/tutorials.

[84] Journal Of L A T E X Class and Files. Rényi divergence and kullback-leibler divergence. 6 (1), 2007. URL https://arxiv.org/pdf/1206.2459.pdf.

[85] Ziteng Sun, Peter Google, Ananda Theertha, Suresh Google, H Brendan, and Mcmahan Google. *Can You Really Backdoor Federated Learning?* URL https://arxiv.org/pdf/1911.07963.pdf.

[86] Brittany Shammas. A man walked down a street with 99 phones in a wagon. google maps thought it was a traffic jam., Feb 2020. URL https://www.washingtonpost.com/technology/2020/02/04/google-maps-simon-weckert/.