École Normale Supérieure Paris-Saclay


Faculty of Science

# Personalized federated learning

Fine tuning and aggregation

Master Thesis in Machine Learning

*Author:*
Mohammed Hssein

*Supervisors:*
Marie Garin
Argyris Kalogeratos

*Referee:*
Michael Arbel

October 2022

# Contents

# Acknowledgment

I would like to thank my supervisors Marie Garin and Argyris Kalogeratos who helped me during this internship. It could not have been accomplished without their support, insightful comments and hard questions. Their remarks helped me during all the time of the internship.

Besides my internship supervisors, I would like to thank my academic referee Mr Michael Arbel, and all my teachers at École Normale Supérieure Paris-Saclay in the MVA master for their advice, encouragement and immense knowledge. I would like to thank especially my deep learning and sequential learning teachers for their complete courses.

# Notations and basic notions

Before starting, we present here a list of mathematical notations we use along this report. **Probability distributions** are often denoted in capital letters like $\mathcal{Q}, \mathcal{P}, \mathcal{D}$ ...etc. When we consider samples from a particular distribution $\mathcal{D}$, we often denote the set of these samples as $\widehat{\mathcal{D}} = \{x_1, ...., x_n\}$ and we call it the **empirical distribution**. **Expectations** $\mathbb{E}$ are always computed with respect to a probability measure, and measure the mean of a random variable. Expectations are often denoted by $\mathbb{E}_{x\sim\mathcal{D}} f(x)$, which means that we compute the expectation of the random variable $f(x)$ where $x$ is sampled from $\mathcal{D}$. Empirical distribution of cardinal $n$ contains samples $x_1, ..., x_n$ that have been drawn from their theoretical distribution in an i.i.d manner, thus we also denote their arithmetic mean by $\mathbb{E}_{x\sim\widehat{\mathcal{D}}} f(x) = \frac{1}{n}\sum_{i=1}^{n} f(x_i)$.

In machine learning and statistical learning theory, we work with functions that measure the quality of predictions. These measurable functions are called **loss functions**. Loss functions are denoted by $l$. We assume that the loss function that we work with $(x, y) \to l(x, y)$ is bounded, and $\forall x, y\ 0 \leq l(x, y) \leq 1$. In addition, the loss functions we consider are assumed to be **convex** with respect to the first variable. From any loss function, we can define a **loss functional** that we denote by $\mathcal{L}$ or $\mathcal{L}_{\mathcal{D}}$, to reference a distribution. Loss functionals are defined over a set of measurable functions (predictors, machine learning models) called **hypothesis set** and denoted here by $\mathcal{H}$. We have for each $h \in \mathcal{H}$ : $\mathcal{L}_{\mathcal{D}}(h) = \mathbb{E}_{x,y\sim\mathcal{D}} l(h(x), y)$. In this work, instead of the notation $(x, y) \sim \mathcal{D}$ that means that $x, y$ is a sample from the distribution $\mathcal{D}$ where $x$ represents the vector of features and $y$ its true label, we will assume there exist a function $f$ such that for each $x$ the number $f(x)$ designs the true label of $x$. We then always write $x \sim \mathcal{D}$ and the loss functional becomes $\mathcal{L}_{\mathcal{D}}(h) = \mathbb{E}_{x\sim\mathcal{D}} l(h(x), f_{\mathcal{D}}(x))$ instead of $\mathcal{L}_{\mathcal{D}}(h) = \mathbb{E}_{(x,y)\sim\mathcal{D}} l(h(x), y)$. Depending on the context, we can denote $f$ also by $f_k$ to designate user $k$, ... etc.

Following the notion of loss functional, comes the notion of **risk**. In statistical learning, with convenient hypothesis on loss function the minimization problem $\min_{h\in\mathcal{H}} \mathcal{L}_{\mathcal{D}}(h)$ has at least one solution. If we denote $h^*$ one such solution, then $\mathcal{L}_{\mathcal{D}}(h^*)$ is called the **true error** or theoretic error. Using optimization techniques (first order techniques for example), one can approximate a solution $\widehat{h}$ of the previous problem using samples $\widehat{\mathcal{D}}$ from the distribution $\mathcal{D}$, i.e $\widehat{h} = \operatorname{argmin}_{h\in\mathcal{H}} \mathcal{L}_{\widehat{\mathcal{D}}}(h)$. We then call the difference $\mathcal{L}_{\widehat{\mathcal{D}}}(\widehat{h}) - \mathcal{L}_{\mathcal{D}}(h^*)$ the **empirical risk** in this report.

Finally, we denote by $\mathcal{H}_i$ for some integer value $i$, the set: $\big\{x \to l(h(x), f_i(x)), h \in \mathcal{H}\big\}$. This set will intervene a lot in the notion of **Rademacher complexities**. Rademacher complexity, is a quantity that measures the learning capacity of a hypothesis set. Given some samples $\widehat{\mathcal{D}} = \{x_1, ..., x_n\}$, and i.i.d Rademacher random variables (random variables $\epsilon_k$ with values only in {-1, +1} each occurs with probability 0.5), Rademacher complexity is defined as:

$$\widehat{\mathcal{R}}_{\mathcal{D}}(\mathcal{H}_i) = \mathbb{E}_{\epsilon}\Big[\frac{1}{n}\sup_{h\in\mathcal{H}}\sum_{k=1}^{n}\epsilon_k l(h(x_k), f_i(x_k))\big|x_1, ..., x_n\Big].$$

This usually is referred to as an experimental Rademacher complexity. Taking expectation with respect to $x_1, ..., x_n$ would result to the theoretic Rademacher complexity. The last notion we define is a pseudo-distance (though we call it distance by abuse of language), that would play an important role in measuring the distance between two distributions based on a hypothesis set $\mathcal{H}$. We call it **discrepancy distance**, and it is defined as:

$$\operatorname{disc}_{\mathcal{H}}(\mathcal{Q}, \mathcal{P}) := \max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{P}}(h) - \mathcal{L}_{\mathcal{Q}}(h)\Big|.$$

When the hypothesis set is known, we could also index this distance using the loss function used. It verifies the **triangular inequality** and the **positivity** assumptions. The last thing we hope to add is the number of data of each user $i$ is denoted either by $m_i$ or $n_i$.

# 1 Introduction

Machine learning have known tremendous developments in the last century. With the development of advanced techniques in many areas, machine learning has become an essential pillar in everyday life. More and more smartphone apps integrate machine learning models to perform various tasks encountered in users' everyday life as object recognition, next word prediction, keyboard sentence completion and machine translation, ... etc. It is theoretically proven, and experimentally validated, that the quality of a machine learning model's prediction is correlated, among many factors, to the quantity and quality of data used for the training. This is perhaps, why in any machine learning model life-cycle, maintenance is required, and is often done by retraining the model as data in practice suffer from **distribution shift**.

Another major issue faced by machine learning in practice, is the problem of **security**. Many users raise many concerns about sharing their data with a central server in order to get an updated version of their apps (models). In addition, sharing data, is not always as simple as we could think of. Indeed, imagine hundreds of thousands of users sharing each hundreds of images of high quality. This would put a huge pressure on the internet traffic, not to mention the **privacy** concerns in areas where sharing data is strictly forbidden as for instance in health sector. These considerations among others have led to the development of a new branch of machine learning called **Federated learning**.

**Federated learning** is a new branch of machine learning, or a machine learning setting, introduced the first time by Google [3] in 2016 - 2017. In this setting, multiple **nodes** or **clients** collaborate in solving a machine learning problem under the coordination of a **central server**. Each client's raw data is stored locally in the node side, and is not transferred of exchanged. Instead, other parameters, are shared as for example the model's parameters, gradients .... etc. The role of the central server is to organize the learning process only. The crucial idea brought by federated learning is to rely on **local training**, i.e the training of local models in each node / client. The federated learning workflow can be summarized in four steps: first, nodes get the global model from the central server. This is the **initialization** step. Second, local nodes train this model using local data. This step is usually referred to as **local training**. Then, the nodes (*all the nodes, sometimes only a portion of the nodes. See Appendix 1*) must update the central server by sending back information to it (sending back their models, gradients, ... etc). This is the **update** step. The forth step, is called the **aggregation** step where the central server aggregates the information received from local nodes and sends back the updates to these nodes. This cycle is called a **communication round** and can start again once the last step is finished. In real life, not all the nodes are available anytime to communicate with the server, because of internet connection for example, and only a portion of the nodes, noted $K$ is available. These are called the **hyper-parameters** of the federated setting. Usually, one finds two major frameworks in federated learning, **cross-device federated learning** for example used in IoT or mobile apps, and **cross-silo federated learning** which implies the use of federated learning in medical of financial institutions. In the cross device framework, usually the number of nodes varies from thousands to millions, with only a portion available to communicate each time. In the second framework, we have only tens to hundreds of nodes available each time for communication. For more details about these two frameworks, please refer to Appendix 1.

The aggregation step, is perhaps the most challenging in federated learning. In early 2017, two famous algorithms were published [3, 4], **FedAvg** for federated averaging, and

**FedSGD** for federated stochastic gradient descent. FedAvg, consists on averaging parametric models obtained from nodes in a point-wise manner, i.e parameter by parameter, whereas FedSGD consists on averaging the gradients. More details about these algorithms, can be found at [3, 4]. A question is still even currently under investigation, that is what is the **best way to aggregate models**. Usually, a convex combination of weights is used, for example the canonical mean, i.e each client gets a weight of $\frac{1}{N}$, where $N$ is the number of participating nodes. A weighted mean is used too, in which we consider some sort of importance to accord to each user, depending on how much data he possess, i.e each user gets the weight $\frac{n_i}{\sum_{i=1}^{N} n_i}$, where $n_i$ the number of data the user $i$ owns. In theory, this is equivalent to minimizing a weighted combination of losses. However, in theory, it is not always clear why aggregating information this way, would improve the performance of some given user $i$. In addition, the federated learning setting, is still facing many other challenges to overcome, for example:

- **Privacy and security**: surely, in this setting, nodes do not share their data. However, to the best of my knowledge, it is not known how much the information carried by model parameters or gradients, with the number of data points, could be sensitive, especially if the model architecture is known in advance.
- **Communication costs**: In **One-shot federated learning**, where only one round of communication is performed, the communication cost is small, but in many cases, for example in vision problems, we need many rounds of communication for some extra large models (ResNets, Inception, ... etc).
- **Data heterogeneity**: Regarding data, there are two scenarios in federated learning. The first scenario, is when users have the same distribution of data, i.e no matter which user we chose to participate in the training, the samples from that user are drawn from a common (universal) distribution $\mathcal{D}$. This is an easy case, as the model trains from one source no matter which user communicates with the server. Another difficult, and more realistic scenario, is when each user has its samples drawn i.i.d from a particular distribution $\mathcal{D}_k$ that is specific to this user. Training a global model in this case is not an easy task. Furthermore, this scenario, needs much more rigorous mathematical framework, by introducing metrics that measure the similarity between distributions. That is why, more and more research is focused on **personnalization** [9]. Personnalization means, to adapt/personnalize model to fit the characteristics of each node. We will discuss this in more detail in the next section.
- **System heterogeneity**: The most challenging factor in the deployment of federated architectures, is system heterogeneity, which means possible issues of network bandwidth, asynchronous Internet connections, ...etc that may affect the learning process.

In this internship, we focused on **how much could we benefit from the knowledge captured by a global model, and the local nodes, to improve the performance of a specific node**. The first section of this report is an introduction. The second section exposes state of the art research conducted in this area. In the third section, we show a statistical learning study and expose the theoretical guarantees we obtained for generalization. In the forth section, we illustrate the theory, with some experiments, and we leave the proofs of propositions and lemmas to the appendix.

# 2 Related work

## 2.1 Personalization

Many factors in practice force the data that the users own to be highly non-i.i.d. For example, in a supervised regression setting, an app that can predict the price of a house given some geographical, structural ... information and that was trained on houses in a given country, would definitely fail to give relevant predictions in another country. Another common example in classification, is the example of sentence completion, known also as next word prediction using google keyboard for instance. The situation could be, given the sentence "*I live in* ", to predict the city where the user lives. This would differ depending on the user's context and use. The consequence of this is that a global model trained on a specific dataset, would definitely fail in a different context. We need a way to adapt the model to its new context of use, i.e we need to **personalize** models. Such an approach, would be useful in many applications, with natural infrastructure to deploy an adapted models to each user, especially in large scale scenarios as in federated learning.

There are many works of personalization in federated learning. A line of work, established connections between **multi-task learning**, **meta-learning** in the context of federated learning. Smith et al [11] studied the problem of multi-task federated learning, and proposed an algorithm that jointly learns the similarity matrix between tasks as long as the models' parameters. Liang et al [10] proposed to store some layers of the model locally, and to train the rest with the federated learning setting. Another line of work, is proposed by Mohri et al [9] who proposed three approaches for personalization with applications to federated learning.

The first approach from [9], called **users clustering**, consists on clustering users into groups of similar users based on a similarity measure of the distributions. Their idea, is that local model could be subject to **over-fitting**, while global models cannot generalize well in various contexts. Clustering users into similar groups would be an intermediate solution. Furthermore, they proved a theoretical guarantee by showing that if users are clustered into groups $\mathcal{C}_1, ...\mathcal{C}_q$, then with probability at least $1-\delta$:

$$\sum_{k=1}^{p} \frac{m_k}{m}\Big(\mathcal{L}_{\mathcal{D}_k}(\hat{h}_{f(k)}) - \min_{h\in\mathcal{H}} \mathcal{L}_{\mathcal{D}_k}(h)\Big) \leq 2\sqrt{\frac{2p\log\left(\frac{2q}{\delta}\right)}{m}} + \sqrt{\frac{dqe\log\left(\frac{m}{d}\right)}{m}} + \sum_{k=1}^{p}\frac{m_k}{m}\text{disc}(\mathcal{C}_{f(k)}, \mathcal{D}_k)$$

where **disc** is the **discrepancy pseudo-distance** defined in notations at the beginning of the report. The last term highlights the distribution mismatch between clusters. Their algorithm is called **HypCluster**. One of the shortcomings of this result is that the data communication cost that users have to exchange in order to form clusters, is $q$ times higher than for training for one model. In addition, the value of the hyper-parameter $q$ is purely experimental and there is no way to approximate it directly unless via some hyper-parameters tuning.

The authors of [9], proposed another approach called **data interpolation**. The core idea of this approach, comes from **transfer learning** (Section 2.2), which is another machine learning field. Since we focus on personalization, the user's data is the most important. However, this data might be small in quantity, and thus to avoid over-fitting, we need to use other data from a reference distribution $\mathcal{C}$ so the resulting distribution would be $\lambda\mathcal{D}_k + (1-\lambda)\mathcal{C}$ with $\lambda \in ]0,1[$ a hyper-parameter. Their algorithm, called **Dapper**, starts from a initial (central) model, and then for each client, uses a grid search over the values of $\lambda$ using a cover of $[0,1]$ to find the best model. They showed then that if $\mathcal{H}$ is the hypothesis set with a VC dimension of $d$, then with probability at least $1-\delta$, the **statistical risk** is

bounded by:

$$\mathcal{O}\left(\sqrt{\frac{\lambda^2}{m_k}+\frac{(1-\lambda)^2}{m_{\mathcal{C}}}}\sqrt{d\log\frac{1}{\delta}}\right)$$

where $m_{\mathcal{C}}$ the number of data points in the dataset $\mathcal{C}$. This apporach, comes with many shortcomings. First, the proposed algorithm does not start from zero, but from an already trained (or at least well pre-trained) model and the choice of the distribution $\mathcal{C}$ is not clear. Second, the algorithm requires $\mathcal{O}\big(P\times\mathcal{A}\times t\big)$ to converge with $\mathcal{A}$ being the cardinal of a cover of the interval $[0,1]$ and $t$ time required to train one model. The communication cost with the server is already very high.

A third approach, is called **model interpolation**. Deep and huge model require lots of data to be well trained. The idea of this approach, is to say instead of interpolating data, one can start from a pre-trained model and try to find an optimal interpolation of all the models for each user/node. More fomally, the quantity to minimize is the following:

$$\min_{h_c\in\mathcal{H}_c,(\lambda_1,\ldots,\lambda_p)\in[0,1]^p,(h_{l,1},\ldots,h_{l,p})\in\mathcal{H}^p}\sum_{k=1}^{p}\frac{m_k}{m}\mathcal{L}_{\mathcal{D}_k}((1-\lambda_k)h_c+\lambda_k h_{l,k})$$

The authors proposed an EM-like algorithm to perform such a task called **Mapper**. The statistical risk, is bounded, with probability at least $1-\delta$, by:

$$2\sqrt{\frac{\log\left(\frac{1}{\delta}\right)}{m}}+2L\left(\sqrt{\frac{d_c}{m}\log\left(\frac{em}{d_c}\right)}+\sqrt{\frac{d_l}{m}\log\left(\frac{em}{d_l}\right)}\right)$$

For more details, please refer to [9]. At this level, one should mention that, finding the global model $h_c$ and local models $h_{l,k}$ in this approach is not an independent task. Mapper, jointly learns the two types of models at the same time.

Finally, Sivek et al [13], gave another framework for a particular type of personalization in federated learning known as **Agnostic Federated Learning**. In this framework, the central model is optimized for any target distribution formed by a **mixture** of of nodes distributions'. Furthermore, they show that this framework, hods intrinsically, a notion of **fairness**. In the next subsection, we expose another essential component in our work, which is **transfer learning**.

## 2.2 Transfer learning

The problem of federated learning, is closely related to another learning scenario where a certain mismatch between the source distribution and a target distribution is considered. This naturally includes the problem of **domain adaptation** or **transfer learning** from a source domain to a target domain [8, 7, 2].

**Transfer learning**, is an area of machine learning that studies how we can store gained knowledge obtained from learning to solve a certain problem, and use it to solve another, usually similar problem. For example, We could try to adapt a model that has been trained to recognize trucks, in cars recognition. The original task is related to a known distribution, we call **source distribution**, and the task we aim to solve later, is related to another distribution we call **target distribution**, we can refer to transfer learning this time also as **domain adaptation**. One advantage of transfer learning, is its ability to transmit knowledge in situation where learning could be very hard, for example when the number of data points at hand is very small, even in the case of non-labeled test targets [2]. Many modern

machine learning tasks belong to this category of situations, for example in computer vision (object recognition, points-cloud compression, ...etc) or in natural language processing in which domain adaptation is a systematic approach (Transformer based architectures like Bert, GPT, ...etc).

Federated learning could be seen, in the context of personalization, as an application of transfer learning, in fact as $P$ problems of transfer learning, $P$ being the number of active nodes. The source target is composed of the $P$ sources, and the target is one of these sources each time. Typically when considering a transfer learning or domain adaptation approach, two main steps are considered:

- **Model pre-training**: this step consists on giving some sort of a warm-up to the machine learning model we consider. Usually machine learning models start from a random configuration of parameters, and as the training with respect to a task continues, the weights start to converge to some minima points convenient to solve the task. The pre-training step, cuts the training in a certain point before it over-fits the initial task. Yet, this is not the only way to perform pre-training. In neural architectures, a typical approach is to train a model completely for the first task, and then to change the last layer/layers to adapt the architecture to the desired target task in next steps.
- **Fine tuning**: this is the second step that follows pre-training. The idea is to start from a pre-trained model, and train it in few iteration on the target distribution. In practice, there are several ways to perform such task. we focus on neural networks, as they are the state of the art model in the majority of machine learning tasks nowadays. It is well known that neural networks are universal approximates [6]. They are a stack of neural layers of various types (convolutions, residual connections, LSTMs, ...etc). The last layer is usually considered as a linear layer used either for classification of regression. During pre-training, the first layer of neural architecture learns the deep features that characterize the task at hand and work as **feature extractors** as for example convolutions in vision tasks [14]. Fine tuning, can be done as:
  - **normal fine tuning**: as we described before. for example in NNs, by fine tuning only the last linear layer using small learning rates, with few iterations.
  - **end to end fine tuning**: this approach, consists on fine tuning the entire model. One drawback is that fine tuning the feature extractors may destroy the information learned. Thus, in practice, usually, we start first by a normal fine tuning, and then by an end to end fine tuning with an extremely small learning rate.

# 3 Contribution

In this section we show and discuss our results. We begin by introducing our starting point, then propose mathematical guarantees based on a statistical learning point of view. We conclude this section by a discussion of these results and the possible potential investigations. We leave experimental evaluation to the next section.

## 3.1 Personalization via aggregation of models

In this theoretical part, we present one approach of personalization that is the personalization via the aggregation of models. In practice however, we add another possibility of personalization that is the fine-tuning. Aggregation of local models is performed using fine-tuned models. The first theoretical result is based on the discrepancy distance.

### 3.1.1 Discrepancy distance

We have seen in the previous section, that personalization is required when users' domains could be different. Model aggregation [9] is a possible way to achieve this goal. However, in that formulation (Section 2.1), the global model and local ones are not obtained in an independent manner, which makes it difficult to use in large scale in practice due to communication costs and privacy (Section 2). User clustering 2.1 suffers from worse computational drawbacks too. Our idea, is to mix the two previous techniques. We will consider fine tuning to transfer the knowledge gained from a global federated model, and try the aggregation approach by finding for each client the best convex weighting scheme. But first let's forget about fine tuning and focus only on model aggregation.

Formally, we dispose of $P$ nodes, each owns a set of samples $\widehat{\mathcal{D}}_k$ with cardinality $m_k$ coming from distribution $\mathcal{D}_k$, $k = 1, ..., P$ Consider a loss function $\mathcal{L}$, and let $h_k = \text{argmin}_{h\in\mathcal{H}}\mathcal{L}_{\mathcal{D}_k}(h)$ with $\mathcal{H}$ being a hypothesis set. the empirical estimators are $\widehat{h}_k = \text{argmin}_{h\in\mathcal{H}}\mathcal{L}_{\widehat{\mathcal{D}}_k}(h)$. The models we try to study for each client $k$ are of the type:

$$\bar{\widehat{h}}_k = \sum_{i=1}^{P} \lambda_i^k \widehat{h}_i$$

where $\lambda_i^k \in [0, 1]$ such that $\sum_{i=1}^{P} \lambda_i^k = 1$. We start with a decomposition of the risk:

**Lemma 3.1.** *Risk decomposition. See appendix 5.2 for proof*
*we have for each user $k = 1, ..., p$*

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_k}(\bar{\widehat{h}}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k) \leq \lambda_k^k\Big(\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k)\Big) + \sum_{i=1, i\neq k}^{p} \lambda_i^k \Bigg[ &\underbrace{\Big(\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i)\Big)}_{term\ (2)} \\ &+ \underbrace{\Big(\mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(h_i)\Big)}_{term\ (1)} \\ &+ \underbrace{\Big(\mathcal{L}_{\mathcal{D}_i}(h_i) - \mathcal{L}_{\mathcal{D}_k}(h_k)\Big)}_{term\ (3)} \Bigg]\end{aligned}$$

The following lemmas will play an important role in the proof of our theorem.

**Lemma 3.2.** *Rademacher concentration. See appendix 5.2 for proof*
*We have with probability at least* $1-\delta$ *over the set of all samples of size* $m_k$*:*

$$\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_k}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\right| \leq 2\widehat{\mathcal{R}}_{\mathcal{D}_k}(\mathcal{H}_k)+3\sqrt{\frac{\log(2/\delta)}{2m_k}}$$

**Lemma 3.3.** *Self discrepancy concentration. See appendix 5.2 for proof*
*Suppose we have a bounded loss* $l \leq M$ *where* $M>0$ *is a constant. The discrepancy between a distribution* $\mathcal{Q}$ *and its empirical version* $\widehat{\mathcal{Q}}=\{x_1,...,x_m\}$ *satisfies:*

$$disc_l(\mathcal{Q},\widehat{\mathcal{Q}}) \leq 2\widehat{\mathcal{R}}_{\mathcal{Q}}(L_{\mathcal{H}})+3M\sqrt{\frac{\log(2/\delta)}{2m}}$$

*with probability at least* $1-\delta$ *over the set of samples of size* $m$*.*

And, the last result is:

**Lemma 3.4.** *Mutual discrepancy concentration. See appendix 5.2 for proof*
*The following inequality:*

$$disc_l(\mathcal{P},\mathcal{Q}) \leq disc_l(\widehat{\mathcal{P}},\widehat{\mathcal{Q}})+\left(\widehat{\mathcal{R}}_{\mathcal{P}}(L_{\mathcal{H}})+\widehat{\mathcal{R}}_{\mathcal{Q}}(L_{\mathcal{H}})\right)+2\left(\sqrt{\frac{\log(4/\delta)}{2m}}+\sqrt{\frac{\log(4/\delta)}{2n}}\right)$$

*occurs with probability at least* $1-\delta$

### 3.1.2 Results

We start this section by our first result, and leave the proof for Appendix 5.2.

**Theorem 3.5.** *Risk high probability bounds*

*we have with probability at least* $1-\delta$*:*

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_k}(\widehat{\widehat{h}}_k)-\mathcal{L}_{\mathcal{D}_k}(h_k) \leq 2 & \underbrace{\sum_{i=1,i\neq k}^{p}\lambda_i^k disc_{\mathcal{H}}(\widehat{\mathcal{D}}_i,\widehat{\mathcal{D}}_k)}_{\text{Term (1) : Purely experimental}} \\ & \underbrace{+2(1+\lambda_k^k)\widehat{\mathcal{R}}_{\mathcal{D}_k}(\mathcal{H}_k)+6\sum_{i=1,i\neq k}^{p}\lambda_i^k\widehat{\mathcal{R}}_{\mathcal{D}_i}(\mathcal{H}_i)}_{\text{Term (2) :Can be estimated}} \\ & \underbrace{+10\sqrt{\log(4p/\delta)}\Big(\sum_{i=1,i\neq k}^{p}\frac{\lambda_i^k}{\sqrt{2m_i}}+\frac{2}{5}\big(1+\frac{1}{2}\lambda_k^k\big)\frac{1}{\sqrt{2m_k}}\Big)}_{\text{Term (3) :Constant term}}\end{aligned}$$

This result establishes an upper bound of the empirical risk as a high probability claim. This bound contains terms of various types:

- Term (1) illustrates the distribution mismatch between users. This dissimilarity is measured here with the discrepancy metric over the set of empirical estimators $\widehat{h}_i, i=1,...,p$
- Term (2) contains empirical Rademacher complexities, and illustrates the learning capacity of the models. This term is a sum of expectations, and can be estimated in practice.

- Term (3) controls the sharpness of the bound, and illustrates the role of number of data points of each user.

The weights appear in the upper bounds which is good news as it makes it possible to **personalize** for each model the weights so that the bound can be minimized. We note here that the dissimilarity between distributions is captured by the discrepancy distance. One could achieve similar results by defining other kinds of distances. This the purpose of the next sections.

## 3.2 Another similarity distance

We have seen in the previous subsection, a way to capture, dissimilarity between distributions, which is the discrepancy distance [9]. In this subsection, we define another distance we call **sim distance** that would lead to similar results, i.e similar high probability bounds of the empirical risk as in Theorem 3.5. To do this we have to put some additional assumptions on the loss function $l$. We suppose the loss function $l$ satisfies the following points:

- $l$ is symmetric, i.e $l(x, y) = l(y, x)$ for all $x, y$
- $l$ satisfy the **triangular inequality**, that is: $l(x, y) \leq l(x, z) + l(z, y)$ for all $x, y, z$

The distance we define here computes the similarity between hypotheses elements $f$ and $g$.

**Definition 3.6.** sim distance
Let $\mathcal{D}$ a probability distribution. Define a distance between $f$ and $g$ as:

$$d_{\mathcal{D}}(f, g) = \mathbb{E}_{x\sim\mathcal{D}}\Big[l(f(x), g(x))\Big]$$

and its **empirical version** for some samples $\widehat{\mathcal{D}} = \big\{x_1, ..., x_m\big\}$

$$d_{\widehat{\mathcal{D}}}(f, g) = \mathbb{E}_{x\sim\widehat{\mathcal{D}}}\Big[l(f(x), g(x))\Big] = \frac{1}{m}\sum_{i=1}^{m} l(f(x_i), g(x_i))$$

The following are some properties satisfied by this distance.

**Lemma 3.7.** *Distance concentration. See appendix 5.2 for proof*
*fix some $f \in \mathcal{H}$. We have, for each $h \in \mathcal{H}$, with probability at least $1 - \delta$:*

$$d_{\mathcal{D}_k}(f, h) \leq d_{\widehat{\mathcal{D}}_k}(f, h) + 2\mathcal{R}_{m_k}(\tilde{\mathcal{H}}_f) + \sqrt{\frac{\log(1/\delta)}{2m_k}}$$

*with the notation $\tilde{\mathcal{H}}_f = \big\{x \to l(h(x), f(x)), h \in \mathcal{H}\big\}$*

### 3.2.1 Results

We have this result very similar to the previous theorem 3.5:

**Theorem 3.8.** *Risk high probability bound. See appendix 5.2 for proof*
*we have with probability at least $1 - \delta$:*

$$\mathcal{L}_{\mathcal{D}_k}(\widehat{\widehat{h}}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k) \leq \underbrace{\sum_{i=1}^{p} \lambda_i^k d_{\tilde{\mathcal{H}}}(\widehat{h}_k, \widehat{h}_i)}_{Purely\ experimental}$$
$$+ 8 \underbrace{\left( \left(1 - \frac{\lambda_k^k}{2}\right) \widehat{\mathcal{R}}_{\mathcal{D}_k}(\mathcal{H}_k) + \sum_{i=1, i\neq k}^{p} \lambda_i^k \widehat{\mathcal{R}}_{\mathcal{D}_i}(\mathcal{H}_i) \right)}_{Can\ be\ estimated}$$
$$+ 12\sqrt{\log(2p/\delta)} \underbrace{\left( \sum_{i=1, i\neq k}^{p} \frac{\lambda_i^k}{2\sqrt{m_i}} + \frac{2 - \lambda_k^k}{2\sqrt{2m_k}} \right)}_{Constant\ term}$$

*with the following notation:* $\mathcal{H}_i = \{x \to l(h(x), f_i(x)),\ h \in \mathcal{H}\}$

### 3.3 Discussion

The previous results reflect various factors related to the task we have at hand. Both theorems come with an probabilistic upper bound that contains a term that illustrates the distribution mismatch between nodes, a term that illustrates the leaning capacity of the hypotheses set, and a term showing the effect of the number of samples per user. However this theoretical results, does not capture many things we have used in our experiments. Recall that we are considering two possible levels of personalization. One level consists on a simple fine-tuning operation and the other level takes into account also the weighting scheme.

The first gap in the theory, is on the structure of the hypotheses sets we are considering. Our machine learning models considered are convolutional neural networks. In fine-tuning step, we keep the feature extractors (the convolution part), and train only the linear layer on top. This means that the sets we are considering are of the form $g \circ \mathcal{H}$ on test time. This would change **Rademacher complexities**, and even the values of distances that measure similarity between distribution. The second gap in this theory, is that personalization via weights, depends upon the weighting scheme. This weighting scheme should be chosen to minimize the upper bounds, but then a question emerges: are these bounds optimal? To answer this question, a possible way would be to find similar lower bounds which is not an easy task. Another challenge is that the upper bound is a linear expression of weights. The minimization of such term lead to putting all the mass on one point (the combination of the three terms is minimal) and zero elsewhere. But this might be far from personalization via model's aggregation since we would end up considering only one model. But since there is a part of randomness in this bounds carried out by discrepancies and Rademacher complexities, this is a randomized version of the **portfolio optimization problem** [1]:

$$\begin{array}{ll} \min\limits_{\lambda} & \sum_{i=1}^{p} \lambda_i a_i \\ \text{s.t.} & \sum_{i=1}^{p} \lambda_i = 1 \\ & \lambda_i \geq 0 \end{array}$$

The randomized version is presented for example in [5]. Please refer to Appendix5.3 for more details. In practice however, this is not a very interesting weighting scheme, as it requires to estimate $\mathcal{R}_{m_i}(\mathcal{H}_i) + \text{disc}_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_k)$, which means each user must have access to all the other distributions (samples).

# 4 Experiments

This section is dedicated to the experimental evaluation of our work. In the first section, we explain the datasets we used and the machine learning models we considered. Before to conclude, we show the experimental results we obtained.

## 4.1 Datasets and machine learning models

To conduct our experiments, we used a **synthetic dataset** and **FEMNIST** (federated extended mnist) dataset both present in LEAF [12]. LEAF, is a open-source benchmark for federated learning. It consists on open-source datasets, statistical metrics, and reference implementations to implement and evaluate federated architectures. The two datasets we used, are classification datasets, the first one with 5 classes, and the second one consists of 62 classes. **FEMNIST** or federated extended **MNIST** consists on $28 \times 28$ gray-scale images of handwritten digits, small letters and capital letters. One advantage of FEMNIST, is that the use of the first 10 classes only, yields a new dataset with is the classic MNIST of handwritten digits. An incredible advantage of working with LEAF, is that it allows to create many different situations using one dataset. For example, it allows to chose the number of users to use, how many samples per user, whether to consider the i.i.d case or the non-i.i.d ...etc.

For the machine learning models, we have used neural architectures. For the synthetic dataset, a simple neural network with only one layer was sufficient to get good results. However, for the second dataset, as it consists on a vision problem, we used a convolutional neural network architecture. It is made of a block of convolutions containing two layers of convolutions with different number of channels and pooling layer, which makes it a feature extractor block, and finally a linear layer for classification.

## 4.2 Experimental evaluation

Many experiments are conducted to test the idea of personalization via fine-tuning and aggregation. Recall, that we work under the non-i.i.d assumption which is a more challenging and realistic situation. However, in real life, another scenario could often happen, where some users have only data that belongs to some class or category and other users for whom data belongs to other categories. This is known to be as a **heterogeneity** problem. In addition, in real life, we may a face a situation where only portion of users participate to training due to internet connection problems, whether users devices are active or shut-down, ...etc. All the experiments we do are conducted in two situations: in the case of non-heterogeneous case and in the heterogeneous case. At this level, we mention that in the non-heterogeneous case, most of the users have most of the categories, but not all. for more details, please refer to Appendix 5.4.

Each experiment, consists on training a global model based on the collaboration of all the users/a fraction of the users in a federated way. The algorithm we have chosen for this purpose is **FedAvg**, where each user $k$ participates in the update by a weight of $\frac{m_k}{\sum_{i=1}^{p} m_i}$ where $m_i$ denotes the number of data points in the training each user $i$ owns. After this step of global training, we proceed to the fine-tuning step, where we fine tune the global model to each user. For the synthetic dataset, since the model is only one linear layer, we fine tune this layer with no other choice. For all experiments using FEMNIST, we do not fine tune the entire architecture, but only the last linear layer. This step is the first type personalization. The last step, is to consider the model $\widehat{\widehat{h}}_k := \sum_{i=1}^{p} \lambda_i^k \widehat{h}_i$ and evaluate its performance. This step, could lead to a second step of personalization, with the weights $\lambda_i^k$ chosen properly for each user $k$. Please refer to Appendix 5.4 for more details.

We can observe from the experimental results (Section 5.4) that relying on purely local models when quite some quantity of data is available is not a good idea, and thinking of personalization via fine-tuning is a systematic approach when possible. Fine-tuned models' performance is comparable to the performance of the global model when trained centrally. In the federated learning setting and in non-heterogeneous case, there is a drop of performance of 0.5% to 4% in contrast to what we expected. A question raises here perhaps about the hyper-parameters used in fine-tuning. However, in the heterogeneous case, fine-tuned models perform well compared to global models (5% to 32% better). Personalization via fine-tuning and model aggregation, does not work well all the times. This perhaps is due to the choice of the optimal weighting scheme that is difficult to find. We note that at this moment of redaction of this manuscript, the weighting scheme obtained via 5.3 has not been fully tested yet, however, we can clearly observe that its use has increased the performance of the aggregated models in 9, 11 and 13. But overall, personalizing via fine-tuning seems better that personalizing with model aggregation. In all the experiment, one can notice that as long as we increase the number of classes, the performance degrades a bit. This is natural since the task gets more complicated. The feature extractors, are not capable of distinguishing many classes that share the same characteristics with the initial ones. The figures 7 and 6 illustrate this fact. They represent embedding of vectors obtained after the convolution operation. The algorithm we used to get these embeddings is TSNE (time distributed stochastic neighbor embedding).

## 4.3 Conclusion and perspectives

This internship has been a valuable opportunity to test the idea of personalized fine tuned models aggregation. The theoretical part of this internship with proofs contains many ideas that can be used differently and lead to other, perhaps more sharper upper bounds, since we do not know how much these bounds we obtained are optimal, as we have no theoretical similar lower bounds. This work however was not a waste of time. Finding a weighting scheme that minimizes these upper bounds has lead to the increase of the performance of personalized aggregated fine-tuned models. For the practical side, the question of heterogeneity of data is a turning point. Indeed, fine tuning, or aggregation techniques have worked relatively well only in the heterogeneous case. Yet, this is not a clear-cut conclusion. The experiments require some important time to run, which makes it very hard to average the results over many runs and look at the variance. A possible line of research that could be done, would be perhaps to try to model/identify in which case with respect to data or some other factor, personalization via fine-tuning should be considered, rather than personalization via models aggregation and vice-versa.

# 5 Appendix

This appendix contains material not presented in the core of the report. We start by a brief comparison between cross-silo and cross-device federated learning. The next section contains mathematical proofs of lemmas and theorems we presented in section 3. At the end we illustrate characteristics of data and results of the experiments.

## 5.1 Cross silo vs Cross device federated learning

| | Cross device federated learning | Cross silo federated learning |
|---|---|---|
| Example | mobile of IoT devices | medical of financial institutions |
| Data availability | Only a fraction of clients available | All clients available |
| Distribution scale | massively parallel | tens to hundreds of clients |
| Adressability | not accessible | accessible to client ids |
| Client statefulness | stateless | stateful |
| Client reliability | Highly unreliable | relatively few failures |
| Primary bottleneck | connection and communication | computation or communication |

Table 1: Cross-silo vs Cross device federated learning

## 5.2 Proofs

In this section we present the proofs of the lemmas 3.1, 3.2, 3.3, 3.4, 3.7, and theorems 3.5 and 3.8 respectively.

*Proof.* Proof of the lemma about **risk decomposition** 3.1 We have for each user $k$:

$$
\begin{aligned}
\mathcal{L}_{\mathcal{D}_k}(\widehat{\bar{h}}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k) &= \mathcal{L}_{\mathcal{D}_k}(\sum_{i=1}^{p} \lambda_i^k \widehat{h}_i) - \mathcal{L}_{\mathcal{D}_k}(h_k) \\
&\leq \sum_{i=1}^{p} \lambda_i^k \mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_k}(h_k) \\
&= \sum_{i=1}^{p} \lambda_i^k \Big( \mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_k}(h_k) \Big) \\
&\leq \lambda_k^k \Big( \mathcal{L}_{\mathcal{D}_k}(\widehat{h}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k) \Big) + \sum_{i=1, i\neq k}^{p} \lambda_i^k \Bigg[ \underbrace{\Big( \mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) \Big)}_{\text{term (2)}} \\
&+ \underbrace{\Big( \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(h_i) \Big)}_{\text{term (1)}} \\
&+ \underbrace{\Big( \mathcal{L}_{\mathcal{D}_i}(h_i) - \mathcal{L}_{\mathcal{D}_k}(h_k) \Big)}_{\text{term (3)}} \Bigg]
\end{aligned}
$$

This concludes the proof. Note that passing from the first equality line to the second one is allowed thanks to the hypothesis of convexity with respect to the first variable of the loss function. □

*Proof.* Proof of the lemma 3.2 on **Rademacher concentration**

The poof of this results, requires this classic lemma from statistical learning theory, regarding Rademacher complexities of a hypothesis set. With probability at least $1-\delta$, we have:

$$\mathcal{R}_{\mathcal{D}_i}(\mathcal{H}_i) \leq \widehat{\mathcal{R}}_{\mathcal{D}_i}(\mathcal{H}_i) + \sqrt{\frac{\log(1/\delta)}{2m_i}} \tag{1}$$

To prove this fact, one could consider the function

$$h(z_1, ..., z_{m_i}) = \mathbb{E}\Big[\frac{1}{m_i} \sup_{f \in \mathcal{H}_i} \sum_{k=1}^{m_i} \epsilon_k f(z_k) | z_1, ..., z_{m_i}\Big]$$

which satisfies the bounded difference condition with $c_i = \frac{1}{m_i}$, and apply Mc Diarmid's inequality.

In the same manner, one could consider this time:

$$h(z_1, ..., z_{m_i}) = \max_{h \in \mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\Big| = \max_{h \in \mathcal{H}} \Big|\mathbb{E}_{x \sim \mathcal{D}_i} l(h(x), f_i(x)) - \mathbb{E}_{x \sim \widehat{\mathcal{D}}_i} l(h(x), f_i(x))\Big|$$

and remark that $\mathbb{E}h(z_1, ..., z_{m_i}) \leq 2\mathcal{R}_{\mathcal{D}_k}(\mathcal{H}_k)$. It is easy to show that $h$ as defined satisfies the property of bounded differences with the constant $c_i = \frac{1}{n}$, applying Mc Diarmid's inequality yields with probability at least $1-\delta$:

$$\max_{h \in \mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\Big| \leq 2\mathcal{R}_{\mathcal{D}_i}(\mathcal{H}_i) + \sqrt{\frac{\log(1/\delta)}{2m_i}} \tag{2}$$

Now, considering events (1) and (2) each with probability at least $1 - \frac{\delta}{2}$ and summing gives with probability at least $1-\delta$ (the intersection of two events):

$$\max_{h \in \mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\Big| \leq 2\widehat{\mathcal{R}}_{\mathcal{D}_i}(\mathcal{H}_i) + 3\sqrt{\frac{\log(2/\delta)}{2m_i}}$$

which is the desired result. □

*Proof.* Proof of lemma 3.3 on **discrepancy between distribution and its empirical version**

First, let's scale the loss function to $[0, 1]$ by considering the loss $l/M$. We have then:

$$\mathrm{disc}_{\frac{l}{M}}(\mathcal{Q}, \widehat{\mathcal{Q}}) = \max_{h \in \mathcal{H}} \Big|\mathcal{L}_{\mathcal{Q}}(h) - \mathcal{L}_{\widehat{\mathcal{Q}}}(h)\Big|$$

and the previous lemma can be applied, by remarking that $\widehat{\mathcal{R}}_{\mathcal{Q}}(\alpha\mathcal{H}) = \alpha\widehat{\mathcal{R}}_{\mathcal{Q}}(\mathcal{H})$. □

*Proof.* Proof of **mutual discrepancy concentration** lemma 3.4:

One have with probability at least $1-\delta$:

$$\mathrm{disc}_l(\mathcal{Q}, \widehat{\mathcal{Q}}) \leq 2\widehat{\mathcal{R}}_{\mathcal{Q}}(\mathcal{H}) + 3M\sqrt{\frac{\log(2/\delta)}{2m_q}} \tag{3}$$

where $m_q$ the number of samples from $\mathcal{Q}$.

Similarly, with probability at least $1-\delta$:

$$\mathrm{disc}_l(\mathcal{P}, \widehat{\mathcal{P}}) \leq 2\widehat{\mathcal{R}}_{\mathcal{P}}(\mathcal{H}) + 3M\sqrt{\frac{\log(2/\delta)}{2m_p}} \tag{4}$$

where $m_p$ the number of samples from $\mathcal{P}$.

We then remark that, the discrepancy satisfies the triangular inequality, and apply it to $\mathcal{P}$ and $\mathcal{Q}$:

$$\mathrm{disc}_l(\mathcal{P}, \mathcal{Q}) \leq \mathrm{disc}_l(\mathcal{P}, \widehat{\mathcal{P}}) + \mathrm{disc}_l(\widehat{\mathcal{P}}, \widehat{\mathcal{Q}}) + \mathrm{disc}_l(\mathcal{Q}, \widehat{\mathcal{Q}})$$

Now considering events (3) and (4) each with probability at least $1 - \delta/2$, and summing gives the desired result. □

*Proof.* Proof of lemma 3.7 on the concentration of the new distance

Let $f, h$ in $\mathcal{H}$. Define for $S_k = \{x_1, ..., x_{m_k}\}$

$$\phi(x_1, ..., x_{m_k}) = \left| d_{\mathcal{D}_k}(f, h) - d_{\widehat{\mathcal{D}}_k}(f, h) \right|$$

Changing one point changes the function $\phi$ with at most $\frac{1}{m_k}$ thus using McDiarmid's inequality we have:

$$\phi(x_1, ..., x_{m_k}) \leq \mathbb{E}_{S_k}\phi(x_1, ..., x_{m_k}) + \sqrt{\frac{\log(1/\delta)}{2m_k}}$$

now we have:

$$\begin{aligned}
\mathbb{E}_{S_k}\phi(x_1, ..., x_{m_k}) &= \mathbb{E}_{S_k}\left| d_{\mathcal{D}_k}(f, h) - d_{\widehat{\mathcal{D}}_k}(f, h) \right| \\
&\leq \mathbb{E}_{S_k} \sup_{h \in \mathcal{H}} \left| d_{\mathcal{D}_k}(f, h) - d_{\widehat{\mathcal{D}}_k}(f, h) \right| \\
&= \mathbb{E}_{S_k} \underbrace{\sup_{h \in \mathcal{H}} \left| \frac{1}{m_k} \sum_{i=1}^{m_k} l(h(x_i), f(x_i)) - \mathbb{E}_{X \sim \mathcal{D}_k}\left[ l(h(X), f(X)) \right] \right|}_{\text{Use Ghost sample technique}} \\
&\leq 2\mathbb{E}_{X,\epsilon} \sup_{h \in \mathcal{H}} \frac{1}{m_k} \sum_{i=1}^{m_k} \epsilon_i l(h(x_i), f(x_i)) \\
&= 2\mathcal{R}_{m_k}\left( \{ x \to l(h(x), f(x)), h \in \mathcal{H} \} \right) = 2\mathcal{R}_{m_k}(\tilde{\mathcal{H}}_f)
\end{aligned} \tag{5}$$

This concludes the proof. □

Now that we have proven all the useful lemmas, we can move to prove the theorems. The proof, consists on three steps:

- Find an upper bound of each of the terms (1), (2) and (3) with known quantities.
- Use concentration results of random variables that would appear around their means.
- consider the intersection of the events properly and conclude.

Before to move to proving theorems, we will establish the last two technical lemmas, so the proofs of the theorems would look more simpler.

**Lemma 5.1.** *Term 3 bounds*
*The third term in risk decomposition, could be bounded by each term of the following:*

- *equation 3.1*

$$\max_{h \in \mathcal{H}} \left| \mathcal{L}_{\mathcal{D}_k}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(h) \right| + \max_{h \in \mathcal{H}} \left| \mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h) \right| + \left( \mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k) \right) + d_{\mathcal{D}_k}(h_k, \widehat{h}_k)$$

- *equation 3.2*

$$3\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_k}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\right|+\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_i}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\right|+\left(\mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k)\right)$$

- *equation 3.3*

$$disc_{\mathcal{H}}(\mathcal{D}_i,\mathcal{D}_k)$$

*Proof.* Proof
We have the decomposition:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_i}(h_i)-\mathcal{L}_{\mathcal{D}_k}(h_k)=&\mathcal{L}_{\mathcal{D}_i}(h_i)-\mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i)+\mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i)-\mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i)+\mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i)\\&-\left(\mathcal{L}_{\mathcal{D}_k}(h_k)-\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_k)+\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_k)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k)+\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k)\right)\end{aligned}$$

Now the first term $\mathcal{L}_{\mathcal{D}_i}(h_i)-\mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i)\leq 0$ because $h_i=\operatorname{argmin}_{h\in\mathcal{H}}\mathcal{L}_{\mathcal{D}_i}(h)$. We have:

$$\mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i)-\mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i)\leq\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_i}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\right|$$

and similarly

$$-\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_k)+\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k)\leq\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_k}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\right|$$

It results that:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_i}(h_i)-\mathcal{L}_{\mathcal{D}_k}(h_k)\leq&\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_k}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\right|+\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_i}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\right|\\&+\left(\mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k)\right)\\&+\underbrace{\left(\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_k)-\mathcal{L}_{\mathcal{D}_k}(h_k)\right)}_{\text{moving term}}\end{aligned}$$

The last moving term, can be controlled either with:

- sim distance: $d_{\mathcal{D}_k}(h_k,\widehat{h}_k)$
- risk:$2\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_k}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\right|$

these give the two first bounds of the lemma. The last one is straightforward since:

$$\mathcal{L}_{\mathcal{D}_i}(h_i)-\mathcal{L}_{\mathcal{D}_k}(h_k)\leq\mathcal{L}_{\mathcal{D}_i}(h_k)-\mathcal{L}_{\mathcal{D}_k}(h_k)\leq\operatorname{disc}_{\mathcal{H}}(\mathcal{D}_i,\mathcal{D}_k)$$

□

We have a similar result for the term (2) in risk decomposition:

**Lemma 5.2.** *Term 2 bounds*
*The third term in risk decomposition, could be bounded by each term of the following:*

- *Equation 2.1*

$$\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_k}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\right|+\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_i}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\right|+disc_{\tilde{\mathcal{H}}}(\widehat{\mathcal{D}}_k,\widehat{\mathcal{D}}_i)$$

- *Equation 2.2*

$$\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_k}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\right|+\max_{h\in\mathcal{H}}\left|\mathcal{L}_{\mathcal{D}_i}(h)-\mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\right|+d_{\widehat{\mathcal{D}}_k}(\widehat{h}_i,\widehat{h}_k)-\mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i)+\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k)$$

- *Equation 2.3*

$$disc_{\tilde{\mathcal{H}}}(\mathcal{D}_i, \mathcal{D}_k)$$

*Furthermore, the first term in the risk decomposition, can be bounded directly by:*

- *Equation 1*

$$2 \max_{h \in \mathcal{H}} \left| \mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h) \right|$$

*Proof.* Proof
The second term in the risk decomposition is $\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i)$. We have:

$$\begin{aligned}
\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) &= \underbrace{\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_i)}_{\leq \max \ldots} - \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) + \mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_i) \\
&\leq \max_{h \in \mathcal{H}} \left| \mathcal{L}_{\mathcal{D}_k}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(h) \right| + \underbrace{\mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i)}_{\leq \max \ldots} - \mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i) + \mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_i) \\
&\leq \max_{h \in \mathcal{H}} \left| \mathcal{L}_{\mathcal{D}_k}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(h) \right| + \max_{h \in \mathcal{H}} \left| \mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h) \right| + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_i) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i)}_{\text{moving term}}
\end{aligned}$$

The last moving term, can be bounded via many possibilities:

- Using disc distance:

$$\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_i) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i) \leq \mathrm{disc}_{\tilde{\mathcal{H}}}(\widehat{\mathcal{D}}_k, \widehat{\mathcal{D}}_i)$$

- Using sim distance:

$$\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_i) = \mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k) + \underbrace{\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_i) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k)}_{\leq d_{\widehat{\mathcal{D}}_k}(\widehat{h}_i, \widehat{h}_k)}$$

The first possibility gives the first expression in the lemma, and second possibility yields the second expression. As for the last expression, it is quite clear since we can bound directly:

$$\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) \leq \mathrm{disc}_{\tilde{H}}(\mathcal{D}_k, \mathcal{D}_i)$$

This ends the proof. □

Now we can easily prove the theorems. We start with the proof scheme and the adaptation to each case follows immediately.

*Proof.* Theorems 3.5, 3.8, Proof sketch
We propose two among all the possible combinations we could obtain from expressions in lemmas 5.2 and 5.1:

- **1 + 2.2 + 3.2** yields using lemma 3.1:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_k}(\widehat{\bar{h}}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k) &\leq \lambda_k^k\Big(\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k)\Big) + \sum_{i=1,i\neq k}^{p} \lambda_i^k \Bigg[ \underbrace{\Big(\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i)\Big)}_{\text{term (2)}} \\ &\quad + \underbrace{\Big(\mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(h_i)\Big)}_{\text{term (1)}} \\ &\quad + \underbrace{\Big(\mathcal{L}_{\mathcal{D}_i}(h_i) - \mathcal{L}_{\mathcal{D}_k}(h_k)\Big)}_{\text{term (3)}} \Bigg] \\ &\leq 2\lambda_k^k \max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_k}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\Big| + 4\sum_{i=1,i\neq k}^{p} \lambda_i^k \max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\Big| \\ &\quad + 4(1-\lambda_k^k) \max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_k}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\Big| + 4\sum_{i=1,i\neq k}^{p} \lambda_i^k d_{\widehat{\mathcal{D}}_k}(\widehat{h}_k, \widehat{h}_i)\end{aligned}$$

  Thus:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_k}(\widehat{\bar{h}}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k) &\leq 4\big(1 - \frac{1}{2}\lambda_k^k\big) \max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_k}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\Big| + 4\sum_{i=1,i\neq k}^{p} \lambda_i^k \max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\Big| \\ &\quad + 4\sum_{i=1,i\neq k}^{p} \lambda_i^k d_{\widehat{\mathcal{D}}_k}(\widehat{h}_k, \widehat{h}_i)\end{aligned}$$

- **1 + 2.3 + 3.3** yields with lemma 3.1:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_k}(\widehat{\bar{h}}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k) &\leq \lambda_k^k\Big(\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k)\Big) + \sum_{i=1,i\neq k}^{p} \lambda_i^k \Bigg[ \underbrace{\Big(\mathcal{L}_{\mathcal{D}_k}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i)\Big)}_{\text{term (2)}} \\ &\quad + \underbrace{\Big(\mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(h_i)\Big)}_{\text{term (1)}} \\ &\quad + \underbrace{\Big(\mathcal{L}_{\mathcal{D}_i}(h_i) - \mathcal{L}_{\mathcal{D}_k}(h_k)\Big)}_{\text{term (3)}} \Bigg] \\ &\leq 2\lambda_k^k \max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_k}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\Big| + 2\sum_{i=1,i\neq k}^{p} \lambda_i^k \max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\Big| \\ &\quad + 2\sum_{i=1,i\neq k}^{p} \lambda_i^k \mathrm{disc}_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_k)\end{aligned}$$

Focus now for the first bound with **1 + 2.2 + 3.2**. The right hand side contains $p$ terms. According to lemma 3.2, we have for each $i = 1, ..., p$, with probability at least $1 - \delta/p$:

$$\max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\Big| \leq 2\widehat{\mathcal{R}}_{\mathcal{D}_i}(\mathcal{H}_i) + 3\sqrt{\frac{\log(2p/\delta)}{2m_i}}$$

. We know if $A_1, ..., A_p$ are events occurring with probability at least $1 - \delta/p$, then $\mathbb{P}(\bigcap_{i=1}^p A_i) \geq 1 - \delta$. Indeed, by using union bound:

$$\mathbb{P}\Big(\bigcap_{i=1}^p A_i\Big) = 1 - \mathbb{P}\Big(\bigcup_{i=1}^p \bar{A}_i\Big) \geq 1 - \sum_{i=1}^p \mathbb{P}(\bar{A}_i) \geq 1 - \sum_{i=1}^p \frac{\delta}{p} = 1 - \delta$$

Applying this to the each event $i$, yields by summing and rearranging the terms that we have with probability at least $1 - \delta$:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_k}(\widehat{\widehat{h}}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k) \leq & \underbrace{\sum_{i=1}^p \lambda_i^k d_{\tilde{\mathcal{H}}}(\widehat{h}_k, \widehat{h}_i)}_{\text{Purely experimental}} \\ & + 8\underbrace{\left(\left(1 - \frac{\lambda_k^k}{2}\right)\widehat{\mathcal{R}}_{\mathcal{D}_k}(\mathcal{H}_k) + \sum_{i=1, i\neq k}^p \lambda_i^k \widehat{\mathcal{R}}_{\mathcal{D}_i}(\mathcal{H}_i)\right)}_{\text{Can be estimated}} \\ & + 12\sqrt{\log(2p/\delta)}\underbrace{\left(\sum_{i=1, i\neq k}^p \frac{\lambda_i^k}{2\sqrt{m}_i} + \frac{2 - \lambda_k^k}{2\sqrt{2m_k}}\right)}_{\text{Constant term}}\end{aligned}$$

which is the desired result.

Move now to the second bound with **1 + 2.3 + 3.3**. According to lemma 3.4, we have with probability at least $1 - \delta$:

$$\text{disc}_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_k) \leq \text{disc}_{\mathcal{H}}(\widehat{\mathcal{D}}_i, \widehat{\mathcal{D}}_k) + \Big(\widehat{\mathcal{R}}_{\mathcal{D}_i}(\mathcal{H}_i) + \widehat{\mathcal{R}}_{\mathcal{D}_k}(\mathcal{H}_k)\Big) + 2\left(\sqrt{\frac{\log(4/\delta)}{2m}} + \sqrt{\frac{\log(4/\delta)}{2n}}\right)$$

As we have done previously, the right hand side is composed of a sum of $p$ terms, we apply high probability bounds with probability at least $1 - \delta/p$ each time. For the first term:

$$\max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_k}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_k}(h)\Big| \leq 2\widehat{\mathcal{R}}_{\mathcal{D}_k}(\mathcal{H}_k) + 3\sqrt{\frac{\log(2p/\delta)}{2m_k}}$$

for the rest of the $p - 1$ terms:

$$\begin{aligned}\max_{h\in\mathcal{H}} \Big|\mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h)\Big| + \text{disc}_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_k) \leq & 3\widehat{\mathcal{R}}_{\mathcal{D}_i}(\mathcal{H}_i) + \widehat{\mathcal{R}}_{\mathcal{D}_k}(\mathcal{H}_k) + 3\sqrt{\frac{\log(2p/\delta)}{2m_i}} \\ & + \text{disc}_{\mathcal{H}}(\widehat{\mathcal{D}}_i, \widehat{\mathcal{D}}_k) + 2\left(\sqrt{\frac{\log(4p/\delta)}{2m}} + \sqrt{\frac{\log(4p/\delta)}{2n}}\right)\end{aligned}$$

By summing all these terms (considering the intersection of these $p$ events), and using the fact that $2p \leq 4p$, we conclude that:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}_k}(\widehat{\widehat{h}}_k) - \mathcal{L}_{\mathcal{D}_k}(h_k) \leq & 2\sum_{i=1, i\neq k}^p \lambda_i^k \text{disc}_{\mathcal{H}}(\widehat{\mathcal{D}}_i, \widehat{\mathcal{D}}_k) + 2(1 + \lambda_k^k)\widehat{\mathcal{R}}_{\mathcal{D}_k}(\mathcal{H}_k) + 6\sum_{i=1, i\neq k}^p \lambda_i^k \widehat{\mathcal{R}}_{\mathcal{D}_i}(\mathcal{H}_i) \\ & + 10\sqrt{\log(4p/\delta)}\Big(\sum_{i=1, i\neq k}^p \frac{\lambda_i^k}{\sqrt{2m_i}} + \frac{2}{5}\big(1 + \frac{1}{2}\lambda_k^k\big)\frac{1}{\sqrt{2m_k}}\Big)\end{aligned}$$

This is the desired result. □

## 5.3 Portfolio optimization

To introduce the randomness carried by Rademacher complexities (by data at the end), we transform the problem of portfolio optimization:

$$\begin{array}{ll} \min\limits_{\lambda} & \sum_{i=1}^{p} \lambda_i x_i \\ \text{s.t.} & \sum_{i=1}^{p} \lambda_i = 1 \\ & \lambda_i \geq 0 \end{array}$$

to the following problem:

$$\begin{array}{ll} \min\limits_{\lambda} & \sum_{i=1}^{p} \lambda_i \mathbb{E}(x_i) + \lambda_i^2 \mathbb{V}(x_i) \\ \text{s.t.} & \sum_{i=1}^{p} \lambda_i = 1 \\ & \lambda_i \geq 0 \end{array}$$

In our case we have:

$$x_i = \mathcal{L}_{\mathcal{D}_i}(\widehat{h}_i) - \mathcal{L}_{\mathcal{D}_i}(h_i) + 2\text{disc}_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_k) \leq 2 \max_{h \in \mathcal{H}} \left| \mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h) \right| + 2\text{disc}_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_k)$$

we then get:

$$\mathbb{E}(x_i) \leq 2\mathcal{R}_{m_i}(\mathcal{H}_i) + 2\text{disc}_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_k)$$

Now according to the inequality of **Efron-Stein**, as the function:

$$\widehat{\mathcal{D}}_i = \{z_1, ...nz_{m_i}\} \rightarrow \max_{h \in \mathcal{H}} \left| \mathcal{L}_{\mathcal{D}_i}(h) - \mathcal{L}_{\widehat{\mathcal{D}}_i}(h) \right| + \text{disc}_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_k)$$

satisfies the bounded difference property with $c_i = \frac{1}{m_i}$, we have:

$$\mathbb{V}(x_i) \leq \frac{1}{4m_i}$$

We then have to solve that:

$$\begin{array}{ll} \min\limits_{\lambda} & \sum_{i=1}^{p} \lambda_i \underbrace{\left(\mathcal{R}_{m_i}(\mathcal{H}_i) + \text{disc}_{\mathcal{H}}(\mathcal{D}_i, \mathcal{D}_k)\right)}_{\text{To be estimated}} + \lambda_i^2 \frac{1}{8m_i} \\ \text{s.t.} & \sum_{i=1}^{p} \lambda_i = 1 \\ & \lambda_i \geq 0 \end{array}$$

to get a personalized weighting scheme for user $i$.

## 5.4 Experiments

At this level, we give more details about the experiments. We have already presented the datasets we worked with: MNIST, FEMNIST and a Synthetic dataset. Two major settings are considered: **heterogeneous case** and **non-heterogeneous case**. In each case We have four categories of experiments:

- Experiment with 10 classes of handwritten digits (MNIST)
- Experiment with MNIST and 10 other classes of small letters from **a** to **j**.
- Experiment with MNIST, all the alphabet with small letters and capital letters **A, B, C** and **D** with a total number of classes of 40 classe.
- Experiment with entire FEMNIST (62 classes).

The choice of these categories is random. In each experiment, the goal is to compare the performance of several models obtained by several ways of means. The focus of course is upon models obtained via **fine-tuning** and via **weights personalization** (**model aggregation**). Recall that in each experiment, the number of data points in test is larger than the number of data points in train set, which is the context of **domain adaptation** and **transfer learning**. The hyper-parameters we used are described in table 15

#### Personalization via fine-tuning

First, we train local models using data available at each user's level and test the performance of the average model obtained via the classic weighting scheme $\frac{m_i}{\sum_{k=1}^{p} m_k}$. Second, we train a **global model** in a centralized way, as if the data of all clients was available in one server and we test **fine-tuning** to adapt this model to each node, before to test the averaged fine tuned models. Lastly, we repeat the same thing but with a **global model** obtained in federated way via **FedAvg**. The results of these experiments are shown in tables 2, 3, 4, 5 and 6.

#### Personalization via model aggregation

As we mentioned before 3.3, this approach needs more work since there is no certainty about the **weighting scheme** to use. The weighting scheme we have used here is:

$$w_i = \frac{\frac{1}{\mathcal{L}_{\widehat{\mathcal{D}}_i}(\widehat{h}_i)}}{\sum_{k=1}^{p} \frac{1}{\mathcal{L}_{\widehat{\mathcal{D}}_k}(\widehat{h}_k)}}$$

where the loss $\mathcal{L}$ here is the MSE. Due to the lack of time, we have used also the weighting scheme solution of 5.3 but in few experiments (See the captions of the tables). We focused in these experiments on global models obtained with a federated way. As what we described before, we train a federated global model, then we fine-tune it to each node and lastly we try this weighting scheme to test a personalized aggregated model using fine-tuned local models. We perform this steps first with a heterogeneous case, and a non-heterogeneous case. Results can be found in 7, 9, 11, 13

#### Characteristics of data

Before to present the results of the experiments, we start by exposing some characteristics of the data and setting we use. We mentioned in many occasions, that we work under the non-i.i.d assumptions. This means that each user has its own data distribution. The

following pictures illustrate this fact. We choose random users, and display the mean of their data that belongs to one particular class. We can clearly see that the users do not write digits and letters in the same manner.

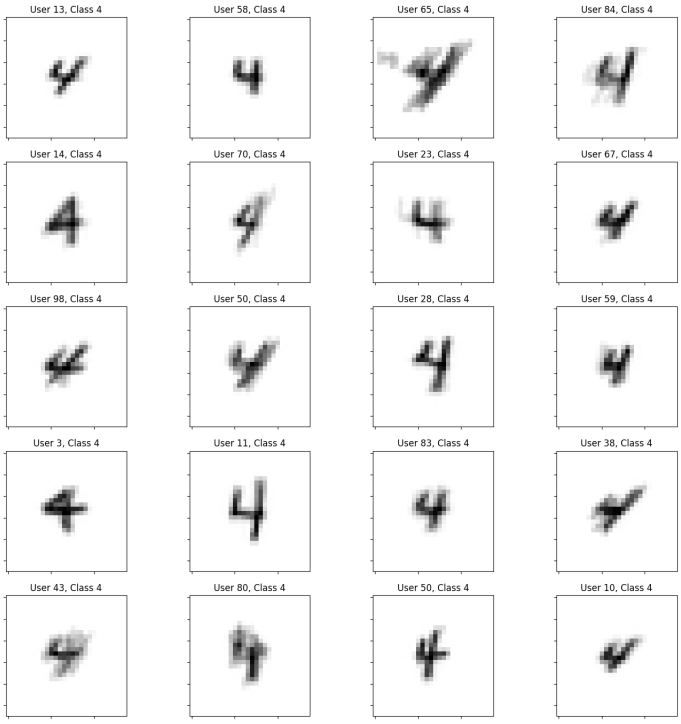


Figure 1: Distribution mean of data that belongs to class 4 of multiple users, data is not i.i.d


Figure 2: Distribution mean of multiple users, data is not i.i.d. Black pictures means that the corresponding user does not have this class in its data

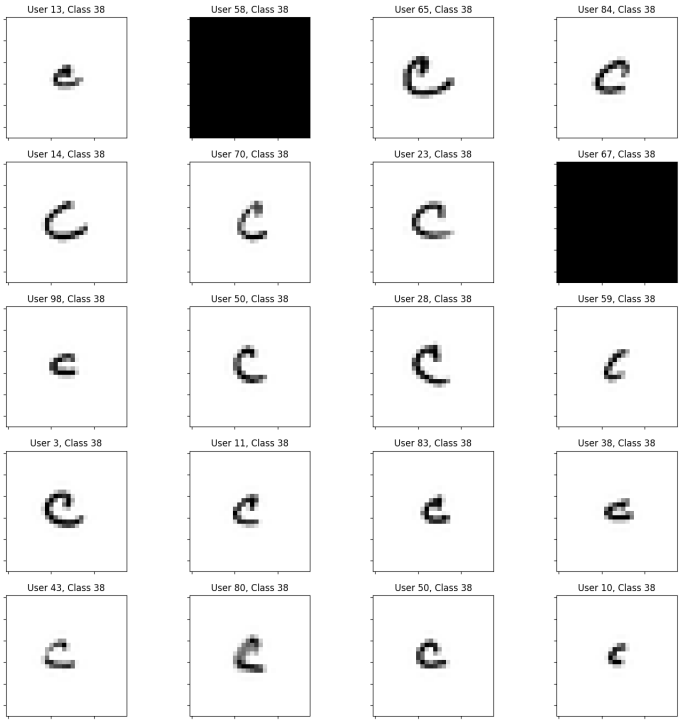


Figure 3: Distribution mean of multiple users, data is not i.i.d. Black pictures means that the corresponding user does not have this class in its data

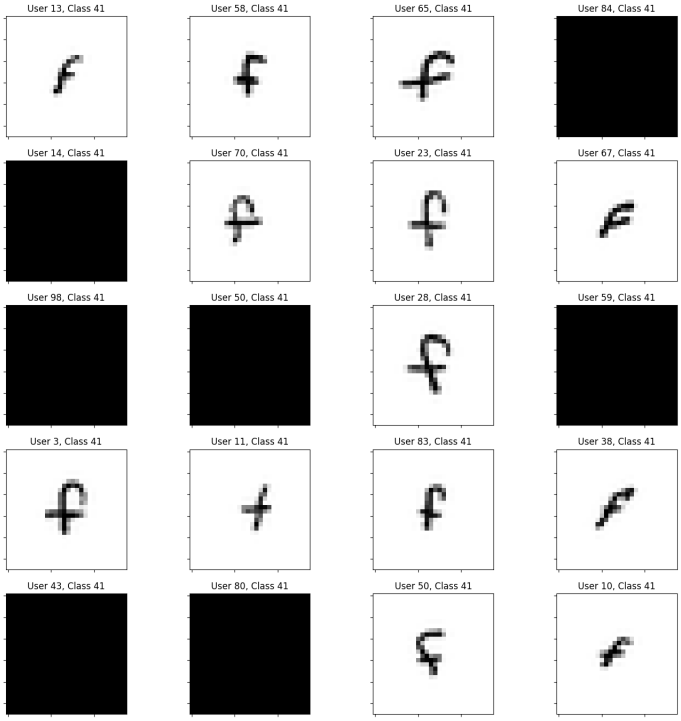


Figure 4: Distribution mean of multiple users, data is not i.i.d. Black pictures means that the corresponding user does not have this class in its data

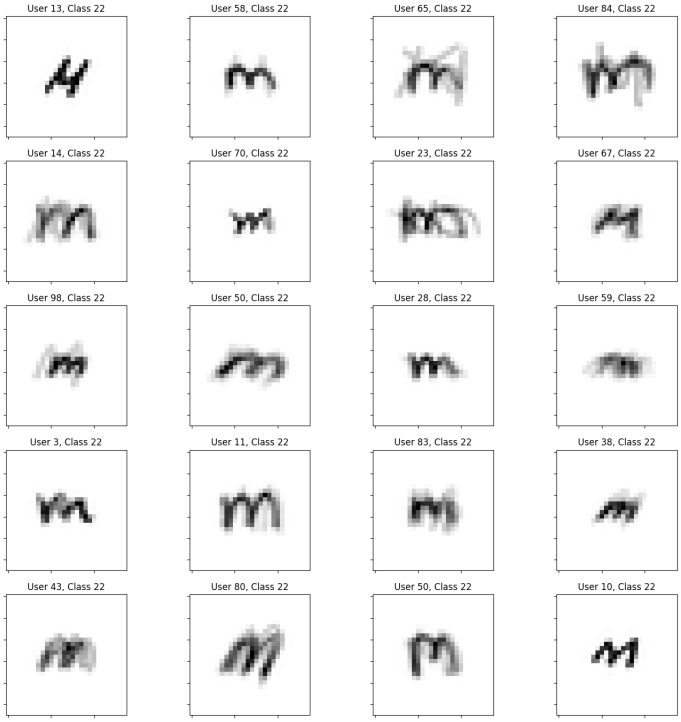


Figure 5: Distribution mean of multiple users, data is not i.i.d

**Personalization : results**

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Local models | 97.18 | 40.57 | 0.95 | 8.09 |
| Averaged Local models | - | 27.82 | - | 2.36 |
| Global model | 98.12 | 98.29 | - | - |
| Fine tuned global model | 99.07 | 98.14 | 0.10 | 0.07 |
| Averaged fine tuned models | - | 98.16 | - | 0.06 |
| Federated Global model | 98.12 | 98.29 | - | - |
| Fine tuned Federated global model | 97.97 | 94.73 | 0.56 | 0.25 |
| Averaged fine tuned models | - | 94.49 | - | 0.26 |

Table 2: MNIST (10 classes of handwritten digits). Note that here averaged models and averaged fine-tuned models do not mean personalization. The weighting scheme used is the same for all the users and it is: $\frac{m_i}{\sum_{k=1}^{p} m_k}$

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Local models | 86.58 | 29.71 | 1.33 | 8.52 |
| Averaged Local models | - | 17.33 | - | 3.55 |
| Global model | 95.33 | 95.58 | - | - |
| Fine tuned global model | 97.54 | 94.09 | 0.17 | 0.20 |
| Averaged fine tuned models | - | 94.33 | - | 0.19 |
| Federated global model | - | 88.74 | - | 0.44 |
| Fine tuned federated global model | 91.65 | 87.06 | 1.23 | 0.56 |
| Averaged fine tuned models | - | 86.25 | - | 0.58 |

Table 3: FEMNIST, Only 20 classes. Note that here, averaged models and averaged fine-tuned models do not mean personalization. The weighting scheme used is the same for all the users and it is: $\frac{m_i}{\sum_{k=1}^{p} m_k}$

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Local models | 90.19 | 30.03 | 1.08 | 8.15 |
| Averaged Local models | - | 9.88 | - | 4.44 |
| Global model | 91.24 | 90.23 | - | - |
| Fine tuned global model | 96.58 | 90.20 | 0.22 | 0.37 |
| Averaged fine tuned models | - | 90.35 | - | 0.36 |
| Federated global model | - | 81.60 | - | 0.66 |
| Fine tuned federated global model | 86.81 | 80.15 | 1.29 | |
| Averaged fine tuned models | - | 79.20 | - | 0.84 |

Table 4: FEMNIST, Only 40 classes. Note that here averaged models and averaged fine-tuned models do not mean personalization. The weighting scheme used is the same for all the users and it is: $\frac{m_i}{\sum_{k=1}^{p} m_k}$

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Local models | 91.86 | 31.26 | 0.99 | 8.42 |
| Averaged Local models | - | 11.00 | - | 4.69 |
| Global model | 85.62 | 83.30 | - | - |
| Fine tuned global model | 94.85 | 84.88 | 0.42 | 0.56 |
| Averaged fine tuned models | - | 85.27 | - | 0.61 |
| Federated global model | - | 75.73 | - | 0.92 |
| Fine tuned federated global model | 83.00 | 75.30 | 1.27 | 1.05 |
| Averaged fine tuned models | - | 74.26 | - | 1.09 |

Table 5: FEMNIST, All classes 62. Note that here averaged models and averaged fine-tuned models do not mean personalization. The weighting scheme used is the same for all the users and it is: $\frac{m_i}{\sum_{k=1}^{p} m_k}$

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Local models | 85.73 | 85.60 | 0.50 | 0.58 |
| Averaged Local models | - | 52.52 | - | 1.44 |
| Global model | 93.40 | 93.24 | - | - |
| Fine tuned global model | 83.52 | 83.56 | 0.54 | 0.42 |
| Averaged fine tuned models | - | 88.87 | - | 0.79 |
| Federated global model | - | 81.41 | - | 0.54 |
| Fine tuned federated global model | 78.75 | 79.36 | 0.93 | 0.76 |
| Averaged fine tuned models | - | 75.95 | - | 0.84 |

Table 6: Synthetic dataset. Note that here averaged models and averaged fine-tuned models do not mean personalization. The weighting scheme used is the same for all the users and it is: $\frac{m_i}{\sum_{k=1}^{p} m_k}$

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Federated Global model | - | 79.74 | - | - |
| Fine tuned Federated global model | 86.80 | 83.71 | 0.63 | 0.92 |
| Personalized avg fine tuned models | - | 80.68 | - | 1.11 |
| Personalized avg fine tuned models 2 | - | 79.38 | - | 1.13 |

Table 7: MNIST (10 classes of handwritten digits), heterogeneous case. The last personalized average model uses the weighting scheme solution of 5.3

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Federated Global model | 98.12 | 98.29 | - | - |
| Fine tuned Federated global model | 97.97 | 94.73 | 0.56 | 0.25 |
| Personalized avg fine tuned models | - | 94.48 | - | 0.25 |

Table 8: MNIST (10 classes of handwritten digits), non-heterogeneous case

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Federated global model | - | 72.72 | - | |
| Fine tuned federated global model | 79.86 | 76.12 | 2.07 | 2.68 |
| Personalized avg fine tuned models | - | 70.55 | - | 3.06 |
| Personalized avg fine tuned models 2 | - | 72.05 | - | 3.20 |

Table 9: FEMNIST, Only 20 classes, heterogeneous case. The last personalized average model uses the weighting scheme solution of 5.3

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Federated global model | - | 88.74 | - | 0.44 |
| Fine tuned federated global model | 91.65 | 87.06 | 1.23 | 0.56 |
| Personalized avg fine tuned models | - | 86.23 | - | 0.58 |

Table 10: FEMNIST, Only 20 classes, non-heterogeneous case

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Federated global model | - | 66.64 | - | |
| Fine tuned federated global model | 74.42 | 68.45 | 3.85 | 4.96 |
| Personalized avg fine tuned models | - | 57.94 | - | 5.90 |
| Personalized avg fine tuned models 2 | - | 59.79 | - | 6.36 |

Table 11: FEMNIST, Only 40 classes, heterogeneous case. The last personalized average model uses the weighting scheme solution of 5.3

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Federated global model | - | 81.60 | - | 0.66 |
| Fine tuned federated global model | 86.81 | 80.15 | 1.29 | |
| Personalized avg fine tuned models | - | 79.30 | - | 0.87 |

Table 12: FEMNIST, Only 40 classes, Non-heterogeneous case

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Federated global model | - | 41.83 | - | |
| Fine tuned federated global model | 72.27 | 61.90 | 4.69 | 6.92 |
| Personalized avg fine tuned models | - | 43.51 | - | 9.22 |
| Personalized avg fine tuned models 2 | - | 49.24 | - | 8.09 |

Table 13: FEMNIST, All classes 62, heterogeneous case. The last personalized average model uses the weighting scheme solution of 5.3. We should mention here that the variance of the performance of the global model is relatively high ($\approx 7.12$), while for the fine tuning federated global model the variance is relatively small ($\approx 1.2$)

| Models | train acc | test acc | train loss | test loss |
|---|---|---|---|---|
| Federated global model | - | 75.73 | - | 0.92 |
| Fine tuned federated global model | 83.00 | 75.30 | 1.27 | 1.05 |
| Personalized avg fine tuned models | - | 73.84 | - | 1.11 |

Table 14: FEMNIST, All classes 62, Non-heterogeneous case

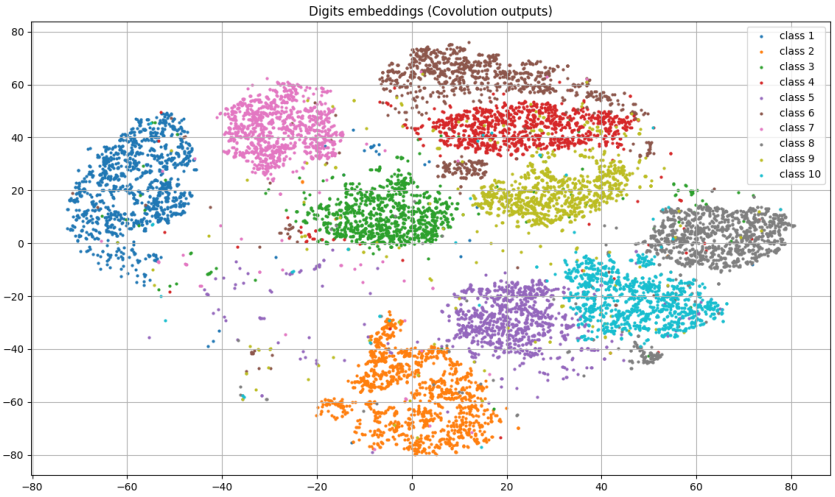


Figure 6: Embedding of feature extractors (convolutions) vectors after global training, MNIST

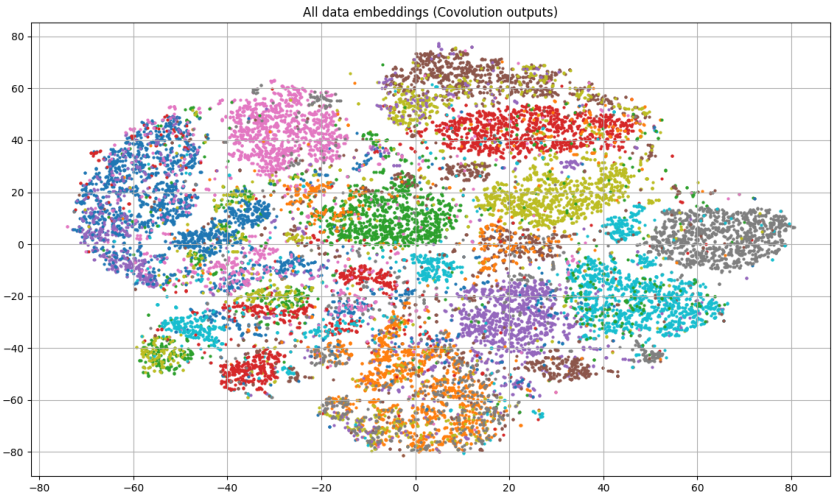


Figure 7: Embedding of feature extractors (convolutions) vectors after global training, FEMNIST

| Hyper-parameter | Value |
|---|---|
| Number of layers (FEMNIST) | 2 conv + 1 pool + linear |
| Number of layers (Synthetic) | 1 linear |
| Convolutions kernel size | $3 \times 3$ |
| Max pooling kernel size | $2 \times 2$ |
| Optimizer for global fed model | SGD |
| learning rate | 0.01 |
| Optimizer for fine tuning | Adam |
| learning rate | 1e-5 |
| Batch size training | 32 (fed) / 64 (fine tuning) |
| Rounds of communication | 60 - 80 |
| Users ratio | 60% - 80% |
| Number of users to train fed model | 300 |
| Number of users used in fine tuning | 200 - 300 |

Table 15: Hyper-parameters

# References


[1] S Boyd and L Vandenberghe. Convex optimization. *Cambridge university press*, 2004.

[2] Guillaume Richard Antoine de Mathelin Georges Hébrail Mathilde Mougeot and Nicolas Vayatis. Unsupervised Multi-Source Domain Adaptation for Regression. *HAL archive ouverte*, 2020.

[3] H. Brendan McMahan Eider Moore Daniel et al. Communication-Efficient Learning of Deep Networks from Decentralized Data. *nternational Conference on Artificial Intelligence and Statistics*, 2016.

[4] H. Brendan McMahan Eider Moore Daniel et al. FedAvg algorithm and fed SGD. *nternational Conference on Artificial In- telligence and Statistics*, 2016.

[5] Marie Garin Theodoros Evgeniou and Nicolas Vayatis. Weighting Scheme for One-Shot Federated Learning. *NeurIPS*, 2022.

[6] Annan Yu Chlo Becquey Diana Halikias Mattew Esmaili Mallory and Alex Townsend. Arbitrary-Depth Universal Approximation Theorems for Operator Neural Networks. *Arxiv preprint*, 2021.

[7] Yishay Mansour Mehryar Mohri and Afshin Rostamizadeh. Domain Adaptation with Multiple Sources. *NIPS*, 2008.

[8] Yishay Mansour Mehryar Mohri and Afshin Rostamizadeh. Domain Adaptation: Learning Bounds and Algorithms. *Google research*, 2009.

[9] Yishay Mansour Mehryar Mohri Ananda Theertha Suresh Jae Ro. Three Approaches for Personalization with Applications to Federated Learning. *Arxiv*, 2020.

[10] Paul Pu Liang Terrance Liu Liu Ziyin Ruslan Salakhutdinov and Louis-Philippe Morency. Think locally, act globally: Federated learning with local and global representations. *Arxiv preprint*, 2020.

[11] Virginia Smith Chao-Kai Chiang Maziar Sanjabi and Ameet S Talwalkar. Federated multi-task learning. *nips*, 2017.

[12] Jakub Konecny H Brendan McMahan Virginia Smith Sebastian Caldas Sai Meher Karthik Duddu1, Peter Wu Tian Li and Ameet Talwalka. LEAF: A Benchmark for Federated Settings. *Arxiv*, 2019.

[13] Mehryar Mohri Gary Sivek and Ananda Theertha Suresh. Agnostic Federated Learning. *Arxiv preprint*, 2019.

[14] Thomas Wiatowski and Helmut Boolcskei. A Mathematical Theory of Deep Convolutional Neural Networks for Feature Extraction. *IEEE*, 2017.