Information Fusion 88 (2022) 263–280

Contents lists available at ScienceDirect

Information Fusion

journal homepage: www.elsevier.com/locate/inffus

Non-IID data and Continual Learning processes in Federated Learning: A long road ahead

Marcos F. Criado [a,*], Fernando E. Casado [a], Roberto Iglesias [a], Carlos V. Regueiro [b], Senén Barro [a]

[a] CiTIUS (Centro Singular de Investigación en Tecnoloxías Intelixentes), Universidade de Santiago de Compostela, 15782 Santiago de Compostela, Spain
[b] CITIC, Computer Architecture Group, Universidade da Coruña, 15071 A Coruña, Spain

ARTICLE INFO

Keywords:
Federated Learning
Data heterogeneity
Non-IID data
Concept drift
Distributed learning
Continual learning

ABSTRACT

Federated Learning is a novel framework that allows multiple devices or institutions to train a machine learning model collaboratively while preserving their data private. This decentralized approach is prone to suffer the consequences of data statistical heterogeneity, both across the different entities and over time, which may lead to a lack of convergence. To avoid such issues, different methods have been proposed in the past few years. However, data may be heterogeneous in lots of different ways, and current proposals do not always determine the kind of heterogeneity they are considering. In this work, we formally classify data statistical heterogeneity and review the most remarkable learning Federated Learning strategies that are able to face it. At the same time, we introduce approaches from other machine learning frameworks. In particular, Continual Learning strategies are worthy of special attention, since they are able to handle habitual kinds of data heterogeneity. Throughout this paper, we present many methods that could be easily adapted to the Federated Learning settings to improve its performance. Apart from theoretically discussing the negative impact of data heterogeneity, we examine it and show some empirical results using different types of non-IID data.

1. Introduction

Machine Learning (ML) consists of the study of mathematical algorithms that improve automatically through experience with the use of data. Traditionally, data used for training ML algorithms are gathered in a centralized dataset, and the process of training can access each data sample at any time. However, in addition to databases, nowadays we live in a society of devices where the main primary computing machines for people in their daily life are smartphones and tablets, equipped with cutting-edge sensors, and computing and communication capabilities. Those devices collect useful data prone to be used for training personalized algorithms that simplify their daily usage. In this context, the quantity of data recorded in just one device may not be sufficient for obtaining an accurate model to perform the desired task. To solve this matter, in the past few years a new paradigm of ML, Federated Learning (FL) [1–3], was developed. This new learning strategy is based on the idea of training a joint model using data from a multitude of coordinated devices in a decentralized way, and has achieved impressive results.

Several problems arise when trying to train a model under these circumstances. For instance, each device has its own processing and storage capacities, which leads to differences in the time needed to perform the training stage [4]. In this paper, we will focus on discussing the statistical variability attached to the use of a myriad of different sets of data, with samples collected in distinct situations. These heterogeneous samples are usually known as *non-IID data* [5], and it is one of the main difficulties encountered in the federated learning process. Assuming data is Independent and Identically Distributed (IID) to avoid some complications, as many works do, is not a good option to deal with real-life situations. Different devices may collect very distinct samples, or even contradictory ones. We will analyse and compare the strategies established so far to face these kinds of issues.

One other assumption in standard ML is that the whole set of samples is available from the beginning of the training stage. However, in realistic tasks, it is frequent to collect data progressively, during several days, or weeks, depending on the context. For this reason, Continual Learning (CL) [6] research gains a lot of importance, since it addresses the difficulties of training a model gradually using real-time collected data, such as variations in data as time passes.

The main difficulties encountered when using CL techniques are *catastrophic forgetting* and *concept drift* [7]. Catastrophic forgetting refers to the phenomenon that occurs when learning a sequence of tasks. In this case, the learning of each new task may cause the model

* Corresponding author.
E-mail addresses: marcos.criado@usc.es (M.F. Criado), fernando.estevez.casado@usc.es (F.E. Casado), roberto.iglesias.rodriguez@usc.es (R. Iglesias), carlos.vazquez.regueiro@udc.es (C.V. Regueiro), senen.barro@usc.es (S. Barro).

https://doi.org/10.1016/j.inffus.2022.07.024
Received 26 November 2021; Received in revised form 13 May 2022; Accepted 28 July 2022
Available online 3 August 2022

to forget the knowledge from previous tasks. Concept drift, on the contrary, is a problem that arises when the model is learning a single task, but the data distribution is not homogeneous. As a result of this, the model performance tends to drop dramatically. These inconveniences occur in any realistic situation that presents a time-evolving nature, such as FL tasks and many others.

In this work, we present some of the possible scenarios that can arise when trying to solve a real problem applying FL, and the difficulties that need to be faced. We classify those scenarios attending to the statistical heterogeneity of data, combining the federated and continual settings to visualize the whole problem, and we present a collection of the most remarkable techniques that have been studied to deal with some of those issues. We also notice that some real situations that involve both Federated and Continual Learning have not been considered nor handled so far, and they should be taken into account [8–10].

The rest of this paper is organized as follows: Section 2 reviews the state-of-the-art techniques for Federated Learning. In Section 3, we present the definition and classification of non-IID data in a federated environment, and we also discuss the different strategies to deal with it. In Section 4, we introduce the Continual Learning framework and the multiple ways data can evolve over time. In Section 5, we combine the different situations of heterogeneous data to show all of the possible scenarios. In addition, we discuss the strategies used to train a model under concept drift that are close to the federated learning framework, and we present a set of restrictions on the data collected that must be verified to apply appropriate strategies. In Section 6 we empirically show how the performance of some strategies drop in heterogeneous settings where the mentioned restrictions are not satisfied. Finally, Section 7 gathers our main conclusions and unsolved challenges.

2. Overview on Federated Learning

Federated Learning (FL) [1–3] is a ML framework used to train an algorithm across multiple decentralized edge devices, where each of them holds its own local data samples. This approach arose in 2016. Federated learning enables multiple clients to build a robust, joint ML model without sharing data, thus allowing to address essential issues such as data privacy and security. The general idea of this technique consists of a loop of local learning stages in the devices, and global parameter aggregations in a central server in the cloud; in such a way that the only shared information are those parameters. There is a model shared by the clients to perform their local training, which is usually a Deep Neural Network (DNN). The most popular federated algorithm is *Federated Averaging (FedAvg)* [11].

There are some key components in standard federated settings. The parameters $w \in W$ of the model are initialized on the server, so that every participant $i \in \{1, \dots, n\} = N$ starts the training stage at a common point. This is crucial for convergence purposes. In each federated round r, a random subset of clients, $i \in N^r \subseteq \{1, 2, \dots, n\}$, of size $|N^r|$, is selected. Those devices receive the current parameter set from the server, w_G^r, perform stochastic gradient descent (SGD) on their local datasets, $D_i \subset X \times Y$, and send back the updated parameters, w_i^{r+1}. Federated learning is typically deployed in a supervised setting, where the data samples $(x, y) \in D_i$ consist of a pair of input data $x \in X$ and its correspondent output $y \in Y$. After performing local training, the local results are aggregated in the server to update the model parameters and start the next round. In the particular case of FedAvg, the local parameters from each client are aggregated applying a weighted mean, to balance the different dataset sizes:

$$\mathbf{w}_G^r = \sum_{i=1}^{N} \frac{M_i}{M} \mathbf{w}_i^r,$$

where M is the total number of data instances at each round, and M_i is the number of instances from client i. After that, a new training round starts, a random subset of clients is selected and the updated model is sent back to those clients.

It is important to notice that most of the works in the literature perform supervised FL, although there are existing works that consider unsupervised settings [12,13]. These works focus on clustering and speech enhancement respectively. In [12], clustering is made minimizing the euclidean distance of the data samples to their corresponding centroid, employing sophisticated techniques such as self-organizing maps to determine the appropriate weights for the weighted mean. On the other hand, [13] develops a method for training a model for speech enhancement using unlabelled data and assuming heterogeneous data distributions across the clients. The case of reinforcement learning has been little studied so far, and there is not a significant volume of works in that area. Consequently, in this review we will only consider supervised and unsupervised strategies.

Another key point in the federated setting is privacy and data security. The distributed spirit of this framework enhances the possibility of training a shared model without actually sharing any piece of information. This is not only desirable but also mandatory. In fact, over the last few years, several governments around the world have implemented data privacy legislation to protect consumers, limiting their data sending and storage only to what is consented by them, and absolutely necessary for processing. Some examples of this are the European Commission's General Data Protection Regulation (GDPR) [14] or the Consumer Privacy Bill of Rights in the US [15]. In addition, a lot of research has been done concerning information sensibility and model threats [16]. For instance, guarantees about the lack of information of the gradient and the neural network weights have been widely studied to preserve client privacy [17]. Even so, the general thought is the Federated Learning framework alone is not enough to keep data privacy [18], and for this reason there are different strategies to obscure the information, such as Homomorphic Encryption [19,20], Differential Privacy [21,22] or Secure Multi-Party Computation [23,24].

However, the training framework we just presented is very recent and innovative, hence presents some challenges that have not been fully addressed yet. For instance, participants selected to train may drop out, or be incapable of performing a local update for the model, due to factors like poor connection or lack of battery. To avoid these issues, there are two options. On the one hand, there are some specific strategies, named Asynchronous FL [25–27], designed to deal with these inconveniences. Devices that check in to the server are required to be plugged in, on a proper connection, and idle, in order to avoid impacting the user of the device. Concerning the convergence of the algorithm, the federated framework presents some shortages. There are no guarantees that training a model under limited communications between local devices and the central server and little computation resources in the devices would lead to a successful result. In fact, there are studies that expose the possible problem of having adversarial participants sending miss-labelled, data-poisoned updates to prevent the model convergence [28]. Apart from adversarial attacks, theoretical and experimental analysis have been carried out on gradient-descent approaches assuming a convex objective loss function [29,30].

Despite being a novel paradigm of ML, FL has attracted a lot of attention. Given the vast amount of research done in the field in the past few years, it became important to keep track of the most remarkable advances. For that reason, it is possible to find reviews on FL that focus on specific issues and show the advances achieved, as well as the remaining open problems. For instance, one of the biggest challenges we already discussed is keeping the clients privacy. This problem is widely studied in [31], where the authors focus not only on privacy guarantees of the federated settings but also on preventing malicious attacks conducted to steal pieces of information. Another interesting review is [4], which concentrates its attention on the problem of communications and the different approaches conceived to perform it. Both of these works also acknowledge the problem of statistical heterogeneity of the data, showing their concerns and exposing the difficulties of working with it. However, little attention is paid to the

actual methods that face this challenge, their advantages and their drawbacks. In this paper, we will focus on those matters.

Statistical heterogeneity of the data is a major issue that needs to be faced in order to construct and deploy a FL model. A bunch of devices training over different local datasets may produce updates in a wide range, thus leading to an undesired result after aggregation, or even worse, impeding the model to converge at all. To prevent these obstacles a common assumption in decentralized learning is considering that the data of the different participants is Independent and Identically Distributed (IID) [3,5,31]. This means that data collected by different participants does not present significant differences. Nonetheless, in most real-life problems and situations this assumption is not satisfied: each client acts in a particular way, thus collects biased data which differs from the one collected by another participant. Further, clients may be interested in applying the global model obtained to predict information in different scenarios. All of these possibilities are gathered under the equivalent concepts of *non-IID data* or *data heterogeneity*, which results ambiguous. Also, in real-world problems it is important to account for another source of heterogeneity: devices constantly extract new data from their environment, and process it sequentially along time, so it is crucial to implement some Continual Learning strategy to adapt the present model to the current data samples [9,10].

There is another difficulty when evaluating heterogeneous data. To test if a method is able to provide the clients accurate models, a dataset that reflects the heterogeneity they present is required. Currently, there are almost no specific benchmark designed to evaluate the goodness of a model that tries to face non-IID realistic problems. One example is the LEAF benchmark [32], which is a framework that include 5 datasets: *FEMNIST, Shakespeare, Sent140, CelebA* and *Reddit*, and reflects domain heterogeneity scenarios. However, there are some other non-IID cases that are not considered, like the ones that affect the clients behaviours. We will further explain these terms shortly. Despite existing such benchmark, most of the works we present in this paper do not employ currently existing benchmarks, and instead they perform their empirical results modifying some datasets to obtain the heterogeneity they require for their experiments. An example of this is the MNIST dataset [33]. Lots of works use this dataset in their experiments, but in some works, images are rotated, different types of noise are added to the samples (domain shift), or the labels of two classes are exchanged with a fixed proportion (behaviour changes). In addition, some modifications had such an impact that they were preserved as distinct datasets, such as MNIST-M [34] or MNIST-c [35]. In the end, each strategy is experimentally tested with a dataset customized in a unique way, distinct from what other works with the same objective do. This is a huge problem as it stands in the way of properly comparing the different methods and analysing which one provides a better result. Establishing at least one common benchmark dataset that represents several kinds of heterogeneous data should be a priority.

3. Non-IID data in Federated Learning

Lots of research has been done regarding the issue of dealing with non-IID data, specially in the context of Federated Learning, where it acquires great importance. In this paper, we will use the words 'heterogeneous data' as a synonym for non-IID data. Existing works focus on both developing new techniques to tackle data heterogeneity [5,36], and proving the convergence of traditional FedAvg trained with non-IID data under some restrictive assumptions [37,38]. However, in most cases, little specifications are made concerning the heterogeneity source of data, if made at all.

Firstly, we are going to give a formal definition of *Independent and Identically Distributed data* (IID data). For that, we need to settle the data probability distributions of the different data owners. Recall we denote a client by $i \in N$, with $N = \{1, \dots, n\}$ being the set of all clients. We also denote each data sample as $d = (x, y) \in X \times Y$, where x is the feature vector of the sample, and y is its corresponding label, in supervised settings. Each client collects its own data D_i, and therefore it has a data probability distribution $D_i(x, y)$, where each data sample (x, y) has probability $P_i(x, y)$. The overall data distribution is a weighted mean of these local data distributions:

$$D_G(x, y) = \sum_{i=1}^{N} \frac{M_i}{M} \cdot D_i(x, y) \tag{1}$$

where M_i is the amount of data collected by the ith device, and $M = \sum_{i=1}^{N} M_i$ is the total number of data samples. Recall that, in this moment, we are working under standard FL assumptions, i.e., the local datasets are fully available from the beginning, so we do not have to deal with shifts in the distributions over time. For that reason, we do not use any temporal index.

Definition 1. Data is said to be IID if the probability belonging to a data sample does not vary as other samples are drawn, and every sample randomly selected can belong to any local dataset with the same probability. In mathematical terms, if we consider the union of the datasets $D = \bigcup_{i=1}^{N} D_i \subset X \times Y$, we claim D is an IID dataset if, and only if,

$$\begin{cases} P_i((x, y), (x', y')) = P_i(x, y) \cdot P_i(x', y'), \\ D_i(x, y) = D_j(x, y). \end{cases} \tag{2}$$

For all $i, j \in N$, and for all $(x, y), (x', y') \in D_i \cup D_j$.

This definition has one important consequence: Under IID data, all the clients distributions are equal to the global distribution:

$$D_G(x, y) = \sum_{i=1}^{N} \frac{M_i}{M} D_i(x, y) = D_i(x, y) \sum_{i=1}^{N} \frac{M_i}{M} = D_i(x, y) \ \forall i \in N.$$

Reciprocally, we say data is non-IID if any of the two conditions given in Eq. (2) are not satisfied. However, this situation is quite less informative than the one from the IID data scenario, since we are unaware of which affirmation is not satisfied, where do the distributions differences lie, etc.

Notice that a local dataset is always IID if the samples it holds are independent. In other words, in settings where the data is centralized or gathered together, data is always identically distributed, since there is only one device involved in training. The term *non-IID* in machine learning implies the existence of various participants, or sets of data, and it is mostly used in the decentralized paradigm.

3.1. Taxonomy of data heterogeneity

Data can be non-IID for many reasons. For instance, there may exist a partition of the clients such that each group presents an IID dataset, but the mixture of them turns out to be non-IID. In fact, this is the reasoning behind the *Group-level personalization* techniques developed in FL (Section 3.2.2). Another option would be that the participants local datasets present slightly different properties while sharing some others. This situation could be handled with *Client-level personalization* strategies (Section 3.2.1). On the whole, the reason for the data to be non-IID is an important piece of information to decide the most convenient approach to face it. Hence, we want to dig deeper into the possible causes of disturbances in the joint probabilities. These causes can rely on multiple elements, and lead to unequal distributions among the clients [31,39]. To characterize the possible causes of non-IID data, it happens to be more useful to think in terms of the probability density functions, $P(x, y)$ and $P_i(x, y)$, rather than distributions, since they can be factorized in two different ways:

$$P(x, y) = P(x) \cdot P(y|x), \tag{3}$$

$$P(x, y) = P(y) \cdot P(x|y). \tag{4}$$

Given these factorizations, we can better distinguish which term represents the clients particularities. If we are dealing with clients who

own data samples unbalanced over the possible classes, the difference between probabilities would lie in the terms $P(y)$ or $P(y|x)$. However, in this work we are going to focus on the factorization given by Eq. (3), since the conditional probability $P(x|y)$ in Eq. (4) may seem controversial with the natural way of training a model, which consists of predicting y based on x and not the other way around.

If the data probabilities belonging to 2 participants, $P_i(x, y)$ and $P_j(x, y)$, are the same, then both factors would be equal: $P_i(x) = P_j(x)$ and $P_i(y|x) = P_j(y|x)$. Notice that every time we say two probability distributions are the same we mean they are alike in statistical terms, i.e., they cannot be recognized as different using a standard hypothesis testing. If, on the contrary, the joint probabilities are not the same, there are three possible scenarios according to the previous factorization:

(i) $P_i(x) \neq P_j(x)$ and $P_i(y|x) = P_j(y|x)$. In this kind of situation, clients own data samples from different domains, but they share the same goal. This could be the case of, for example, participants collecting data for training an autonomous car. Some users may drive on the left and some others on the right, and they will face different circumstances. That will make the input space of the different participants skew. However, they gather data with one common objective, and they are expected to act similarly.

(ii) $P_i(x) = P_j(x)$ and $P_i(y|x) \neq P_j(y|x)$. This sort of scenario occurs when the input spaces perceived by the clients are analogous, but their outputs are not. A real situation where this could happen, related to the training of an autonomous car, is a yellow traffic light. When encountering a yellow traffic light, the correct output for some participants would be to stop the car, and for some others to continue driving without changes. This causes incompatibilities among the clients.

(iii) $P_i(x) \neq P_j(x)$ and $P_i(y|x) \neq P_j(y|x)$. This is a combination of the two previous situations: Participants want to learn a common task, such as driving, but their input spaces are significantly unequal, and their reactions to some of the inputs are different too.

On the whole, this gives us a total of four different situations to account for. We represent them in Table 1, along with the different works that deal with each situation. There are lots of techniques that could fit the cell of IID data, such as FedAvg [11]. However, that situation is out of the scope of our work, and we will focus on the non-IID scenarios. Most of the works that consider heterogeneous data problems do not provide a classification of non-IID data, neither worry about the kind of heterogeneity they are trying to deal with. However, we locate them in Table 1 according to the kind of non-IID data situation that they can solve. For this reason, we establish two possible classifications of the FL non-IID research.

The first way of classifying the different strategies is based on how the works place themselves in the FL context. Most of the FL works that deal with heterogeneous data describe their approach as a *Personalization* approach to increase their accuracy over the different clients. This personalization can be performed at different levels. In Section 3.2 we briefly explain these kinds of methods. It should be noticed that, although these strategies deal with data heterogeneity, they are unaware of which probability density function is varying in each situation.

On the other hand, once we have discussed and explained the different types of non-IID data that could exist, it seems very reasonable to classify the strategies according to the type of non-IID data it faces. This classification encompasses the other one, and at the same time it opens the door to consider other ML techniques that also deal with heterogeneous data and are close to FL. We describe these techniques in Section 3.3.

Table 1
Non-IID learning scenarios in Federated Learning, and the strategies that could potentially solve each situation. Strategies that deal with changes in both the input space and the behaviour are placed only in the last column, and not in the previous ones.

Variations in the clients datasets			
No changes (IID data)	Changes in the input space throughout clients $P_i(x) \neq P_j(x)$	Changes in the behaviour throughout clients $P_i(y/x) \neq P_j(y/x)$	Changes in the input space and behaviour throughout clients
OUT OF SCOPE	**[40—46, 52, 63] [69—75, 78—80] [82—87, 89—93]**	**[47—50] [107—109]**	**[51,53] [58—60]**

3.2. State-of-the-art classification: Personalization strategies

In this section, we focus on the first classification we just mentioned. Some of the works present in the FL literature try to balance the generalized knowledge learned from the whole set of clients with the specificity of each of them. This kind of thinking gives rise to the personalization techniques, which precisely aim to grant more importance to the particular information of individual clients. It is empirically shown that in realistic situations one global model cannot fit the particularities of all clients [37]. In fact, some clients may have opposite interests sometimes, so we must open the door to the possibility of having more than just one global model. Personalization arises as an intermediate agreement between the model generalization and individualization, so that the model can learn not only the general knowledge but also the uncommon one.

Personalization can be implemented at different levels: each participant could have its own model, distinct from all of the others, which we will refer to as *Client-level Personalization*; or there could be groups of clients sharing the same model, i.e., *Group-level Personalization*. Both options present some advantages, as they accomplish a better model performance, although their main drawback is that their computational requirements are more demanding.

3.2.1. Client-level personalization

Client-level personalization refers to the approaches that allow each participant to obtain its own model after the training process. When trying to personalize a global model trained in a distributed setting, a multitude of options and ideas have been studied and proposed. These approaches are gathered into two different classes:

(A) On the one hand, we find previously existent techniques of ML adapted to the FL framework to develop a better global model, such as Transfer Learning or Multi-task Learning.

(B) On the other hand, we encounter different implementations of a simultaneous two-level learning, local and global. Once the global model is obtained, each participant combines their local model, which was trained simultaneously with the global one, and attain their personalized model.

The first type of solutions (*A*) include proposals such as Transfer Learning techniques to adapt pre-trained models over public datasets to a bunch of devices [40,41]. Nonetheless, there are other approaches in this line that search for common representation of data at each client, so that the local updates are resilient against domain shifts, and hence the global model can perform well under non-IID data distributions [42]. There are also adaptations of Regularization Methods, which add a penalization term on the loss function. It is the case of *pFedMe* [43] and *pFedAtt* [44]. The first of them, *pFedMe*, consider a term in the loss function that penalizes very different updates from different clients. Moreover, the resultant loss function is a Moreau envelope, a well-known mathematical object that allows the authors to

provide convergence guarantees and further accuracy analysis. On the other hand, *pFedAtt* introduces a regularization term that augments the contribution of similar updates, grouping analogous clients updates and facilitating convergence in non-IID scenarios where the differences rely on the input distributions. Another idea is adapting algorithms from the Model-Agnostic Meta-Learning (MAML) setting, such as *Reptile*, to a federated setting [45,46]. These works aim to train a model that can easily adapt to different tasks. In order to achieve that, they explore how different data representations affect the model training, and choose the more general representation, because it would adapt faster to any of the goal tasks. There are other approaches similar to these ones, where some authors consider the devices particularities constitute enough difference to assume that the training participants are performing different tasks [47,48], thus bringing up methods based on Multi-task Learning to the FL paradigm, such as MOCHA [48]. All of the above ideas are reinterpretations of what one could understand by personalization, using different points of view to reuse already known techniques.

The latter class of approaches (*B*) focuses on training two distinct models in parallel for each device. One of those two models is the global one, trained jointly by every client with its own data following the standard federated baseline, whereas the other one remains private and will be used to adjust the result from the federated training. The proposals vary, both in the way of achieving the global model and how the private model is taken into account. Here we present four alternatives. In [49,50], clients train a Deep Neural Network (DNN), with the requirement that the last few layers are not shared, and each client trains them separately. In this case, the shared layers play the role of the global model, while these latter layers act as the personalization model, allowing different participants to obtain different results for similar inputs. In [51], clients follow probabilistic steps of training. A certain probability $p \in (0, 1)$ is fixed at the beginning, and in each round of training participants execute locally one step of Stochastic Gradient Descent with probability p, and share the local models of their device to the server with probability $1 - p$. Unlike in common federated frameworks, the global model is computed and distributed to each client, but they do not add the next updates over that model. Instead, they calculate another model using a weighted mean over the global model and their local one, so in the end each client obtains a unique model. On the other hand, the *FedProx* method [52] focuses on training a model similar to FedAvg, but allowing some variation in the different local updates before they converge, and keeping a measure of the updates dissimilarities through the training process. At the end of the training procedure, each participant receives the global model, but they are allowed to modify it briefly according to their dissimilarity measure. To conclude, in [53] devices train a global model as it is done in general in FL, but whenever a client participates in a training round, it trains a local model at the same time it trains the global one. Once training is finished, each client adjusts the global model obtained using the local model.

The experimental results of some of these proposals are quite remarkable. For instance, in [42], authors experimentally show that the global model achieved with their method, trained over the MNIST dataset, also performs well over a rotated version of that same dataset, whereas standard *FedAvg* drops its accuracy significantly. [45] compares their proposed meta-learning method with some other strategies of personalization, such as a fine-tuning baseline [54] and a k-nearest neighbour baseline [55], obtaining an accuracy higher than these two algorithms by over 10%. [49] trains two well-known DNNs, *MobileNet-v*1 and *ResNet*-34, adding some personalization layers. They use the CIFAR-100 and FLICKR-AES [56] datasets, and split the data among clients imposing a non-iid division. They only compare both DNNs with standard FedAvg, however, they highly improve the accuracy, reaching 80% in both datasets, while FedAvg barely gets 60% with CIFAR-100, and 40% with FLICKR-AES. [52] compares FedProx with FedAvg using the MNIST and Shakespeare datasets in both IID and non-IID situations. In the IID scenario, FedAvg performs slightly better. However, FedProx outperforms it by a huge difference in the non-IID setting, and also improves the convergence ratio in both cases. Lastly, [53] uses the MNIST, EMNIST and CIFAR-10 [57] datasets to test the accuracy of their APFL [53] strategy. They obtain an accuracy of 89% over a non-IID split of the dataset, in opposition to FedAvg and FedAvg with fine-tuning, which obtains 32% and 83% of accuracy respectively.

3.2.2. Group-level personalization

In contrast with attaining a distinct model for each client, another alternative is what we call *Group-level personalization*, which consists of gathering the devices in clusters and training a different model for each of them. This line of research emerged a couple of years ago, so it presumably has not achieved its full potential yet. For this reason, there are few different approaches to be mentioned, and all of them are very recent.

The most discussed topic in this area of research is how to split (or assemble) participants in order to get the groups that would benefit the most from sharing a model. One set of approaches are based on adopting hierarchical clustering techniques to partition the clients [58–60]. Their strategy is based on using a measure of distance between the weight updates the clients send to the server to gather [58,59] or divide [60] them. However, there is no mathematical guarantee that participants who send similar updates would benefit each other. It could be the case of some participants who collect very different data but the updates they generate are close to each other. Similarly, nothing ensures that the ones who send different updates would be better off apart. The only justifiable statement is that these participants would reach convergence faster than others, regardless of the performance of the obtained model.

The other approach to perform this kind of personalization concerns the global data distribution, and the local distributions of the clients. The main point of the works concerning this line of research [61–63] is that if data is non-IID among the devices, then a shared global model cannot fit all of the data samples belonging to any client. When this happens, the global distribution may not represent the singularities of some participants, and thus the global model should not be trained according to that distribution. What these works propose is a method to determine the global distribution D_A that best represents the different clients. This distribution does not necessarily coincide with the weighted mean global distribution (Eq. (1)). Clients are then grouped according to some private parameters that depend on D_A. This strategy avoids the usage of the local updates to form the clusters and develops theoretical guarantees to justify that clustering the participants this way benefits the final model.

Concerning the experimental results, [58–60] perform training on MNIST, FEMNIST, and CIFAR-100 [57] benchmark datasets, dividing the samples among the clients and swapping labels in some of them to produce different behaviours, $P(y|x)$. They achieve higher accuracy and convergence ratio than standard FedAvg. However, these methods do not compare themselves with any other algorithm than FedAvg, which is designed to tackle problems only in IID scenarios. On the other hand, [61] improves the result obtained by FedAvg on the task of digit image recognition using the MNIST dataset, in a centralized framework. [62,63] compare themselves with FedAvg in decentralized settings using the datasets of Fashion MNIST and Extended MNIST (EMNIST) [64], and they obtain a similar accuracy. On the whole, the most remarkable improvement accomplished with these kinds of personalization methods so far is their convergence speed. In Table 2 we summarize the datasets used in these works.

3.3. Statistical taxonomy for non-IID strategies

Once we have explained the existing personalization methods in FL, we now want to deepen into the other classification we made, based strictly on the kind of non-IID data the different works face. Going

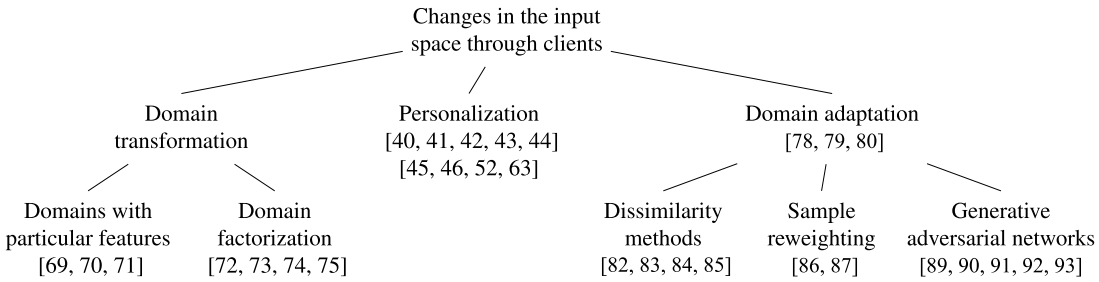


Fig. 1. Classification of the different approaches that are able to solve the problem of spatial heterogeneity in the input spaces.

Table 2
Summary of the datasets employed in the works presented in Section 3.2. Datasets marked with an asterisk are modified in different ways, making it impossible to fairly compare each other. Some of the datasets mentioned were not referenced so far: Omniglot [65], OTB [66], VOT2014 [67], Shakespeare [2] and Fashion MNIST [68].

Article	Datasets used in experiments		
[40]	MNIST;	CIFAR-10	
[45]	MiniImageNet;	Omniglot	
[46]	MNIST*;	CIFAR-10*	
[47]	CASAS		
[49]	CIFAR-100;	FLICKR-AES	
[50]	OTB;	VOT2014	
[52]	MNIST;	FEMNIST;	Shakespeare
[53]	MNIST*;	CIFAR-10*;	EMNIST
[58]	MNIST*;	CIFAR-100*	
[59]	MNIST*;	FEMNIST	
[62]	Fashion MNIST;	EMNIST	
[63]	Fashion MNIST;	EMNIST	

back to Table 1, we classify these methods into 2 categories: those who work with changes in the input space throughout clients, i.e., changes in $P(x)$; and those who work with changes in the behaviour throughout clients, i.e., changes in $P(y|x)$.

3.3.1. Changes in the input space throughout clients

Each participant can collect data from their own input domain. However, in this Section we assume that their domains will remain unchanged during training, i.e., they will follow an IID data distribution over time. In decentralized settings, participants collect data independently, so nothing can assure their input domains are the same although that could be the case for some of them. This is by far the most studied kind of non-IID data, and strategies developed in this line of research pay attention to multiple problems that may occur in real-life scenarios. In this review, we include strategies that are not performed in the FL framework, but that are prone to be adapted to such settings. These works are classified into two main categories: *(i) Domain Transformation* and *(ii) Domain Adaptation*, which also branch into different approaches (see Fig. 1).

Concerning the first of them, Domain Transformation methods focus on detecting the particularities and the common parts of the data domains, and they try to build a new shared input space. After that, the input spaces perceived are transformed into a common input space. As far as we are concerned, this strategy has only been performed in centralized settings. Nonetheless, in a decentralized setting each participant could carry out these calculations and the central server could build the common input space based on all of the local spaces estimated by the clients.

A serious difficulty when trying to deploy this kind of method is that feature spaces in realistic problems tend to be very high-dimensional spaces, and that causes problems such as needing more processing capacity, or inaccurate results due to the curse of dimensionality. For instance, [69–71] consider each domain may have its own set of features to characterize the samples, causing incompatibilities across domains, and they develop methods to extract a common feature representation. A different approach, performed in [72–75], consists of constructing a factorization of the feature space with some properties. [72,73] split the feature space into two orthogonal subspaces: one of them contains the domain variations, whereas the other one keeps the common parts, and both are used separately to perform learning. [74,75] divide the input space into an arbitrary number of low-dimensional spaces and apply techniques of Distance Metric Learning (DML) [76,77] in each of them.

On the other hand, Domain Adaptation methods [78–80] work with a distinct situation, close to Transfer Learning, and they are in general deployed in centralized frameworks. However, some works aligned with this line of research talk about Federated Transfer Learning [81], and consider the FL settings. In these cases, users are assumed to train a model with their data belonging to some domains, but they then apply that same model to get predictions on other domains. In this kind of setting, it is commonly said that the samples drawn for training belong to *Source Domains*, while the samples used for prediction belong to *Target Domains*. The most important difference with the previous category is that in this case there are no training data from the Target Domains, so it is not possible to create an appropriate feature space for learning those domains. Some other works in this line of research [82–85] present a variety of methods to measure the dissimilarity between the source and target domains, such as Maximum Mean Discrepancy (MMD) [82], or Moment Matching for Multi-source DA (M^3SDA) [83] and adjust the training model in an unsupervised manner [84,85].

Another possibility for adapting the domains that also consists on calculating distances among the data distributions, make use of that information for re-weighting the samples from the closest ones to improve the model performance. [86,87] apply this method using the Kullback–Leibler [88] divergence, a well-known metric from information theory.

A different approach for Domain Adaptation is based on Generative Adversarial Networks (GANs) [89–93]. This strategy, that achieves remarkable results, consists of training two neural networks simultaneously: one of them is designed to create fake input data from the different domains, and the other one aims to distinguish the real data samples from the fake ones. Article [93] is particularly interesting because they employ GANs in federated settings. In these works, the method presented is compared with some other pre-existing methods, such as Deep Adaptation Network (DAN) [94], Deep Domain Confusion (DDC) [95] and Residual Transfer Network (RTN) [96]. These are some state-of-the-art techniques in Domain Adaptation. However, GANs methods outperform them in well-known tasks, such as object recognition using the Office-31 [97], Office-Home [98] and VisDA2017 [99] datasets, and digit recognition using the MNIST [33], USPS and SVHN [100] datasets.

In general, comparing the different methods is a tough issue. Each work is free to choose different synthetic datasets to perform experimental results, and also modify them to generate the required heterogeneity that they want to face (see Table 3). For these reasons, it is impossible to fairly compare the diverse strategies we presented. However, there are some remarkable results that we would like to highlight: regarding Domain Transformation methods, the experimental

Table 3
Summary of the Datasets employed in the works presented in Section 3.3.1. Asterisks indicate that the datasets have been modified in particular ways, making it impossible to fairly compare each other. Some of the datasets mentioned were not referenced so far: Bing-caltech256 [102], COREL5000 [103], ImageNet [104] and NYUD [105].

Article	Datasets used in experiments		
[69]	MNIST + SVHN + USPS		
[71]	Office-31;	Bing-caltech256	
[74]	COREL5000;	Trecvid2005[a]	
[75]	MNIST*;	Olivetti FR[b]	
[79]	MNIST + SVHN		
[80]	Office-31;	ImageNet;	VisDA2017
[82]	Office-31;	Image CLEF-DA	
[83]	Digit5;	Office-31	
[84]	MNIST + MNIST-M + USPS;	VisDA2017	
[89]	MNIST + SVHN + USPS;	Office-31;	NYUD
[90]	Office-31;	Image CLEF-DA[c]	
[91]	Office-Home;	VisDA2017	
[92]	MNIST + SVHN + USPS;	Office-31	

[a]Available at http://www-nlpir.nist.gov/projects/trecvid.
[b]Available at http://www.uk.research.att.com/facedatabase.html.
[c]Available at http://imageclef.org/2014/adaptation.

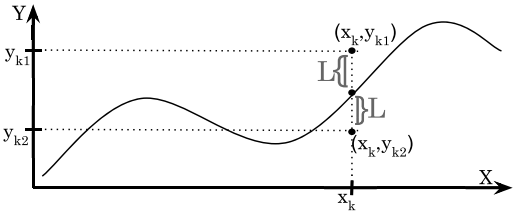


Fig. 2. Regression model in one variable. The samples (x_k, y_{k_1}) and (x_k, y_{k_2}) belong each to one different client. If the distance between those samples is bigger than $2L$, there is no output that could be closer than L for both of them. Hence, at most only one output would be considered correct.

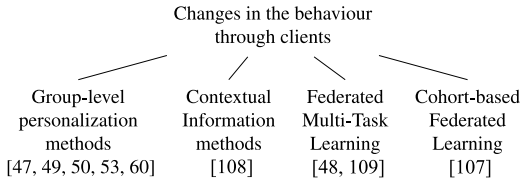


Fig. 3. Classification of the different approaches that deal with the spatial heterogeneity in the behaviours of clients.

results of two of the works stand out [71,75]. They present a complete variety of experiments and contrast their results with other well-known methods, getting significantly better error ratios and accuracies. On the other hand, the most outstanding results achieved with Domain Adaptation methods are the ones from [84,91,92]. The first one, [84] propose their method SimNet and experimentally compare their results with some other methods like DAN, RTN and a baseline method over the datasets of MNIST, Office-31 and VisDA2017. It improves the accuracy obtained by every other method in the three cases. Concerning [91,92], they both employ the Office-31 dataset, and obtain impressive results compared to the other methods they test.

Besides all of the strategies we just talked about, there are a bunch of other methods to deal with the domain shifts. One of them is [101], which also mentions the Source and Target Domains, but also factorizes the input space to search for a Grassmann Manifold that fits all of the data samples. Afterwards, the training is performed only on that manifold, instead of in the whole feature space. Lastly, some of the federated strategies of personalization explained in Section 3.2 can also deal with the kind of heterogeneity brought up in this Section [40,41,45,46,52,63].

3.3.2. *Changes in the behaviour throughout clients*

Differences in the behaviour of the clients refer to discrepancies in their conditional probabilities $P(y|x)$. A variation of this nature means that for, at least for some data samples, the correct output is not the same for all of the clients. More formally:

Definition 2. 2 clients $i, j \in N$ present different conditional probabilities $P(y|x)$ if there exist a significant quantity T of data samples $\{x_k\}_{k=1}^{T}$, and distinct outputs $\{y_{k_1}, y_{k_2}\}_{k=1}^{T}$ such that the participants i, j own the data samples (x_k, y_{k_1}) and (x_k, y_{k_2}) respectively, $\forall\ k \in \{1, \dots, T\}$.

To be precise we need to specify the meaning of "distinct outputs": y_{k_1}, y_{k_2} are distinct if $\|y_{k_1} - y_{k_2}\| > 2L$ for a certain margin of error L (see Fig. 2), which may vary depending on the specific problem. In a classification problem $2L$ needs to be lower than 1, so the margin of error has to be less than $1/2$, whereas in regression problems the margin of error is fixed depending on the level of accuracy desired.

The main problem regarding this context is that, unlike the previous one, a single global model cannot fit all of the users behaviours, since it will not be able to produce different predictions for the same input data. A model in ML is, by definition, a mapping $M : X \longrightarrow Y$ that assigns one value to each possible input [106]. Given the input x_k, a traditional model in a distributed setting would process it equally for all clients, thus predicting one, and only one, output. As a result, the predicted value could be y_{k_1}, y_{k_2} or a different one, possibly intermediate, but in any case one of the clients would obtain a prediction with an error bigger than L. In addition, to detect the existence of several behaviours in the clients, we need to study the result of their loss functions, which implies having a certain amount of labelled data.

To overcome this matter, it is essential to consider some kind of model architecture that grants the possibility of variations among the participants. That is, making it possible for each user $i \in N$ to have a somehow personalized model M_i, distinct from the others. Some of the strategies of personalization discussed in Section 3 can be adapted to deal with different behaviours (see Fig. 3). For instance, having a proper metric of the error would allow the *Group-level personalization* strategies to cluster the participants according to their behaviour. This approach is very similar to Cohort-based Federated Learning [107], which precisely organize clients into cohorts with very similar data distributions.

There are few approaches specifically designed to deal with this kind of non-IID data. One of the closest techniques to tackle this issue, although it does not specifically talk about FL, is the one deployed in [108]. This strategy consists of including additional pieces of information to the input data, such as a task identifier z. This allows 2 very similar data samples x_{k_1}, x_{k_2} to be distinct: $(x_{k_1}, z_1), (x_{k_2}, z_2)$.

To conclude, there are also works in the crossroad between FL and Multi-task Learning [48,109]. The first one was already discussed in Section 3.2. The latter also presents the multi-task framework combined with the federated setting. It performs experiments using the MNIST, EMNIST, and Shakespeare datasets, and compares itself with well-known federated algorithms such as FedAvg and FedProx.

4. Non-IID Data in Continual Learning: Concept Drift

In this section, we are going to consider *Continual Learning (CL)* problems, which involve the training of models over time. In the standard ML setting, the objective is to build a prediction model using a certain amount of data. A key point to discuss is that the training dataset is typically assumed to be fully available from the beginning, and this may conflict with realistic situations, where data is collected progressively and changes over time. For that reason, it is convenient to talk about CL, a ML setting in which models continuously learn and evolve using new streams of data samples, while aiming to retain

preceding concepts. This kind of framework has been given different names over the years [110], like *Lifelong Learning* [111,112], *Never Ending Learning* [113,114] and *Incremental Learning* [115–117], but all of them rely on the same ideas: training a model gradually with data collected over different periods of time, adapting to the new instances and trying to preserve the previous knowledge.

We introduce the CL framework because we aim to talk about the time-evolving condition of FL problems. However, throughout this section we are going to cite and briefly describe works focused on CL that do not necessarily consider the FL framework. This is because, as we already mentioned, there are almost no works that focus on both FL and CL simultaneously [9,10,118]. Nonetheless, the works we consider are, from our point of view, the ones that would be more easily adaptable to the FL framework, with multiple devices collaborating to achieve the same global model. We will further explain how each strategy could be modified when talking about them.

Training a model using CL techniques presents some specific problems, which have already been studied in recent literature. The most challenging ones are, as it occurred with FL, related to the data distribution. CL was conceived as a centralized paradigm of ML so, even though non-IID data across devices has not been discussed nor handled so far, it can evolve in time. This is a complication, as the model could be unable to converge to a solution if the training data shifts constantly. Another undesirable situation, named *catastrophic forgetting*, is that the model completely and abruptly forgets previously learned concepts if they are not present in the current data anymore [119,120]. For these reasons we are going to focus on how data behaves as time goes by, and how to act if the data shifts drastically, in unpredictable ways. This is commonly known as *concept drift* [7,110].

4.1. Concept drift definition

The non-stationary data distribution is caused by changes in data over time. These changes can be seen as variations in the frequencies certain kind of data appears: a concept has frequency zero if it has not appeared yet in the dataset, and when it shows up its frequency becomes a positive number. This kind of variation, called *concept drift*, is one of the most important CL challenges [110,121]. We can formally define them as follows:

Definition 3. Given a time period $[0, t]$, and a set of samples $S^{0,t} = \{(x_j, y_j)\}_{j=0}^{t}$ with a certain probability distribution $D^t(x, y)$, where x_j is a feature vector and y_j is its correspondent output; we say a Concept Drift occurs at timestamp t if there is a significant difference between $D^t(x, y)$ and $D^{t+1}(x, y)$:

$$\exists\, t\, :\, D^t(x, y) \nsim D^{t+1}(x, y).$$

Note that concept drift is a complicated issue, and it becomes even worse in a federated environment. If the problem we are trying to deal with is by nature a federated problem that evolves in time, each client might experiment a drift in different moments. Also, a local concept drift does not necessarily have an impact on the global distribution. It may be the case that a local drift on client i results in a change in the distribution of i, but not in the joint distribution, $D_G^t(x, y)$. In a situation like this, that client should implement some kind of personalization to adapt the global model to its particularities. This is an example of why concept drifts are potentially dangerous for the model performance, and hence must be detected and counteracted.

When trying to deal with concept drifts, one should notice that not all of them are alike, as data can evolve in multiple ways. Similar to what occurred with non-IID data in FL, it is important to characterize concept drifts to distinguish them. However, in the case of concept drifts, most of the existing works present a common ground, and base their classification according to which factor from the equation $P(x, y) = P(x)\cdot P(y|x)$ is altered. According to this criteria, we determine three types of shift [116,122]: (1) virtual, (2) real, and (3) total (see Fig. 4):

(i) *Virtual Concept Drift* makes reference to variations in just the marginal probability density, $P^t(x) \neq P^{t+k}(x)$ and $P^t(y|x) = P^{t+k}(y|x)$. Returning to the example used in Section 3.1 of training an autonomous car, this situation happens, for instance, when clients move into places or regions previously unseen for them.

(ii) *Real Concept Drift*, related to differences in conditional probabilities, $P^t(x) = P^{t+k}(x)$ and $P^t(y|x) \neq P^{t+k}(y|x)$, is caused by a change in the conditional probability of the classes with respect to the input features, i.e., similar input data samples that have unequal labels. An example of this would be, again, a yellow traffic light. Sometimes a client would stop the car when encountering a yellow traffic light, and some others may continue driving.

(iii) *Total Concept Drift* is the mixture of the two other drifts, $P^t(x) \neq P^{t+k}(x)$ and $P^t(y|x) \neq P^{t+k}(y|x)$, and it is the result of both probabilities evolving significantly over time.

This classification is analogous to the one we proposed in Section 3 for the data heterogeneity across clients. Apart from these cases we just discussed, concept drift also takes place when the task itself changes, since that also modifies $D^t(x, y)$. This scenario is closely related to multi-task learning [48,123,124]. Nevertheless, in the scenario we consider the task remains unchanged. It is therefore a single-incremental-task scenario [110].

4.2. Concept drift detection

Once we settled what we understand by concept drift, we can discuss the methods developed to deal with it. Those methods typically consist of three parts. The first step is that they need to detect modifications in the distributions. Then, they have to act in consequence to the detected changes, so the model obtained is adjusted to the current scenario. Finally, it is important to explain the drift and understand their implications for future training. In this Section we just examine the detection strategies. The algorithms that implement a response to these drifts will be reviewed in Section 5.

Many concept drift detection strategies have been proposed to attach the situations of virtual and real drifts [8,125–133]. These approaches are often classified as *Data Distribution-based* or *Error Rate-based* methods correspondingly. They use different statistical properties of the input and output distributions to identify their breaking points, corresponding to drifts. Most of these strategies of concept drift detection consider a situation where data is centralized in one single machine. As far as we are concerned, the only works that present a concept drift detection strategy on federated settings are [8,130]. Nonetheless, the strategies we highlight are, from our point of view, easily adaptable to the FL framework.

Data Distribution-based methods aim to detect virtual concept drift. When trying to detect this kind of drift, the only required information is the input pattern of the data samples, $\{x_i\}_{i=1}^{M}$, or some transformation of it. For instance, the strategy developed in [125] works directly with the input data, and it consists of measuring the similarities among the features, grouping them in clusters, and evaluating the number of features from the new data sample in each cluster to identify a drift. One possible way of adapting this to a federated environment would be that each client calculates their own clusters, and detects local drifts with independence of what occurs for other clients. This drifts would be communicated to the central server, that would be the responsible of taking into account that information.

On the other hand, [126] works with an alteration of the input data. They determine a mapping that relies on the input features of the samples $f : X \subset \mathbb{R}^m \longrightarrow \{-1, 1\}$ and apply it to the whole input dataset, splitting it into two groups (the ones that go to 1 and the ones that go to -1). Then, they statistically compare if data received before and after a certain timestamp is equally distributed in those groups. If they are not, a drift is detected. The map f has to verify certain conditions for this method to work with a high level of precision. One

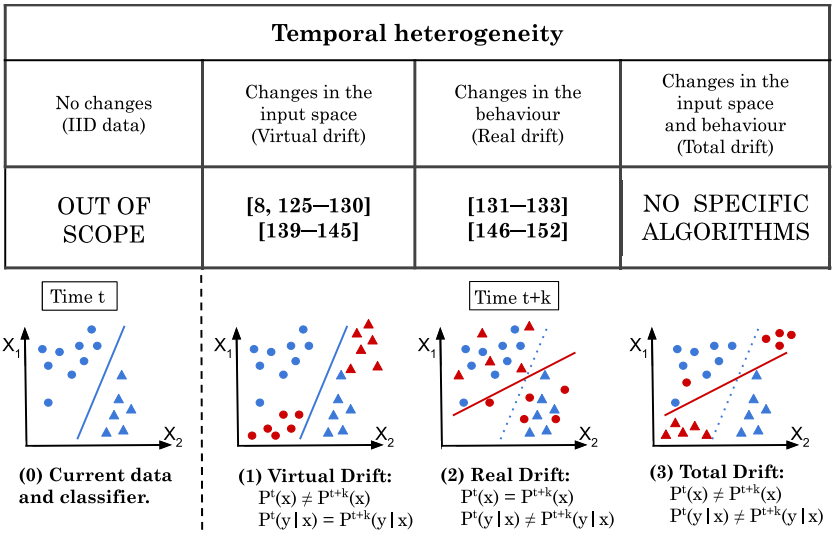


Fig. 4. Representation of Concept drifts in a two-dimensional input space X with two possible labels $Y = \{\circ, \triangle\}$. On the left of the doted line, we find the data samples received before time **t**, and on the right there are three possible time-evolving situations. (1): the new data samples observed are situated in new regions, previously unseen. However, data labels correspond with the split made by the classifier from (0). (2): the new instances appear in already known regions of the input space, but they are incorrectly classified using the model from (0). (3): the two previous situations are combined.

alternative is applying a mapping L from the class of linear functions and then determine the value of f depending on the resulting norm of L. Let $A \in \mathbb{R}$ be a fixed value. Then:

$$L : X \longrightarrow \mathbb{R}^m$$
$$x \longmapsto w^T x, \quad x \in X, \quad w \in [0,1]^m.$$

$$f(x) = \begin{cases} 1 & \text{If } \|L(x)\| \geqslant A \\ -1 & \text{Otherwise} \end{cases}$$

defines a mapping that presents all the required properties. The value of A is chosen according to the values of the features. There are more kinds of possible mappings with good qualities, but these are the simplest ones. To adapt methods like this one to a federated framework the only additional requirement is that all of the clients employ the same map f to perform their calculations, in order to keep the sensibility of the drift detection method equal among the devices.

Another recurrent strategy for the detection of virtual drifts consists of using sliding windows to keep track of the samples received in the past and compare them to the current data stream [8,127–129]. In this kind of method, datasets are split in two according to the time they were collected in, and then the two groups of data samples are compared using different metrics and statistical properties, such as their mean and standard deviation, distances calculated between two samples of the same group, or one sample from each group, etc. Analogous to the previous method, all clients should employ the same metric or statistical parameters to determine whether their data presents shifts or not. However, they do not necessarily have to split their data samples according to the same timestamp, as each client could experience drift in a different moment. This is the strategy developed in [8] in a federated environment.

The other work that presents a strategy to detect virtual drifts in FL frameworks is [130]. This work assumes that for the first stage of training there are no concept drifts, and keeps statistical and numerical information about the updates sent by each client in this stage. After that, the same information is calculated from the next updates of the clients, and the results are compared to the previous ones to determine whether there is a drift.

On the other hand, *Error Rate-based methods* focus on detecting real concept drift. They present more of a challenge in contrast with virtual concept drift detection. In the first place, the virtual concept drift strategies we just presented can be deployed in unsupervised settings since they do not need any label information, whereas real drift detection methods need it because the main variable involved when trying to detect changes in conditional probabilities, $P(y|x)$, is the error in the predictions. Some of these works also employ sliding windows to perform the drift detection [131,132], although in this case the metrics considered must give an especial role to the label information. When the error of the model increases abruptly, a real concept drift is detected. Hence, techniques aiming to detect this kind of variations are highly dependent on how the model inaccuracy is measured [133]. There are different ways of accounting for the model loss. One of the most extended functions for measuring the error in machine learning models is the well-known cross-entropy loss. A lot of research has been made to determine whether this is an appropriate measure of the conducted error [134,135]. In addition, different authors have proposed many other loss functions based on the cross-entropy loss [136–138]. Despite the good results achieved, all of these alternatives present some limitations, such as weaknesses against skew labelled data, or the fact that errors are untraceable. These kinds of properties are very desirable when facing real drift, as they provide important information about the origin of the error.

5. Addressing Federated and Continual non-IID data

For what we have seen in Section 4, concept drift in CL scenarios can be interpreted as the counterpart of *non-IID data* in the FL ones, i.e., changes in the distribution as time passes are the origin of statistical heterogeneity in continual settings. Notice that variations on the distribution of one dataset over time should be always contemplated as identically distributed, since there is only one dataset affected. Nonetheless, the sets of data collected by one client i over time, corresponding to different timestamps, D_i^t, D_i^{t+k}, could be studied as two different datasets, $D^t \subsetneq D^{t+k}$.

In that sense, it is logical to talk about non-identical distributed data over time. In fact, considering again the factorization $P(x, y) = P(x) \cdot P(y|x)$, the casuistry is identical to the one explained in Section 3: if the client data distribution is stationary (IID over time), then we can ensure that both factors remain equal over time: $P_i^t(x) = P_i^{t+k}(x)$ and $P_i^t(y|x) = P_i^{t+k}(y|x)$. Else, we find the three possibilities, illustrated in Fig. 4. From now on, we will call *temporal non-IID data* to the data heterogeneity that a client can undergo over time (see Section 4.1).

Table 4
Spatial and Temporal heterogeneity learning scenarios, and the strategies that could potentially solve each situation. Strategies that deal with changes in both the input space and the behaviour are placed only in the last row/column, and not in the previous ones.

		Spatial heterogeneity			
		No changes (IID data)	Changes in the input space throughout clients $P_i(x) \neq P_j(x)$	Changes in the behaviour throughout clients $P_i(y\|x) \neq P_j(y\|x)$	Changes in the input space and behaviour throughout clients
Temporal heterogeneity	No changes (IID data)	OUT OF SCOPE	**[40–46, 52, 63] [69–75, 78–80] [82–87, 89–93]**	**[47–50] [107–109]**	**[51,53] [58–60]**
	Changes in the input space over time, $P^t(x) \neq P^{t+k}(x)$ (Virtual Concept Drift)	**[8, 125–130] [139–145]**	NOT		
	Changes in the behaviour over time, $P^t(y\|x) \neq P^{t+k}(y\|x)$ (Real Concept Drift)	**[131–133] [146–152]**	ADDRESSED		
	Changes in the input space and behaviour over time (Total Concept Drift)	NO SPECIFIC ALGORITHMS	SO FAR		

Conversely, we will call *spatial non-IID data* to the data heterogeneity across clients that are training a shared model (see Section 3).

In real-life problems, data distributions can vary in a bunch of different ways. Clients in a federated setting are expected to collect their own data samples, under particular conditions, leading to statistically unequal datasets. These differences can rely on the inputs each client perceive, $P_i(x) \neq P_j(x)$, as well as on the label associated with their inputs, $P_i(y|x) \neq P_j(y|x)$ (see Table 1). If we desire the model to be adapted to the particularities of the training participants, standard FL techniques will not be enough. Moreover, the process of collecting data and solving a task takes a certain amount of time, so the desired model should be able to evolve and adjust to future situations. Data will be collected during a long period of time, leading to changes in the input space, $P^t(x) \neq P^{t+k}(x)$ and also in the labels, $P^t(y|x) \neq P^{t+k}(y|x)$ (see Fig. 4).

On the whole, there are 4 feasible scenarios for each spatial and temporal data, and they may appear combined with each other in realistic tasks. The global data distribution $D^t_G(x, y)$, which includes both spatial and temporal heterogeneity, can evolve following 16 different courses. We represent all of the possibilities in Table 4, as well as some of the strategies and algorithms that focus on solving some of those possibilities. Notice that we include IID data to consider all of the possible combinations of heterogeneity.

It is reasonable to think that each course must be faced with specific methods. For instance, if the problem we are considering presents changes in the input space over time across one or multiple participants, it could be solved using a memory-based method to generalize the data from previous distributions and avoid catastrophic forgetting. Despite fitting perfectly for this problem, these kinds of solutions cannot deal with changes in behaviours over time, or changes in the input space across clients. For this reason, in this Section we focus on determining which strategies are more suitable to deal with each circumstance. For now, we only explained some algorithms from FL that were proposed as *Personalization strategies*, without further explanation about the origin of the data heterogeneity; as well as methods to detect drifts, but not to react to them.

Notice that if we determine effective approaches to solve each of the scenarios corresponding to the first row and column in Table 4, then we will be able to solve the situation of any cell by combining the algorithms from its corresponding row (already addressed in Section 3.3.1) and column, as long as they are compatible. Thus, from now on we will consider scenarios where data is IID in the spatial axis. This corresponds to pure CL. In the following sections, we are going to present the existing solutions to deal with temporal non-IID data, classify those strategies according to their shared characteristics, and compare their experimental results when possible.

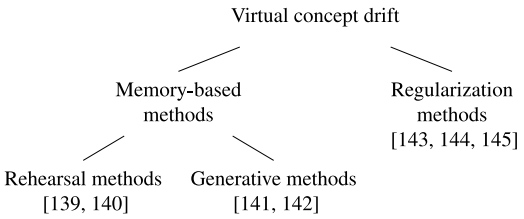


Fig. 5. Classification of the different techniques able to deal with the temporal heterogeneity in the input space of data.

As far as we are concerned, there are only few methods for addressing both federated and continual issues at the same time [9,10]. However, there is still room for significant contributions in these types of scenarios and a lot of situations that have not been taken into account.

5.1. Virtual concept drifts

As we already discussed in Section 4.2, the procedures to detect virtual drifts rely only on the input data distributions. Similarly, the approaches to prevent the model from lowering its accuracy involve just the data samples. They can be classified in *Memory-based methods* and *Regularization methods*, see Fig. 5.

Memory-based methods [139–142] focus on keeping a record of data samples from the previous concepts, so when a drift is detected, the network is trained both with new data and the data recorded to avoid forgetting. [139] proposes a method (CLEAR) to store the data samples already used for training and use them again in the future, combined with the new collected data. They show that this strategy is quite effective to prevent catastrophic forgetting. [140] proposes a similar technique, ER, but with the difference that they just store little amounts of data, and use them repeatedly after drifts mixed with the new samples. They conclude that their approach neither harm generalization nor causes overfitting to the saved data samples.

There are other existing alternatives encased in the Memory-based methods which consist of using the recorded data to generate similar samples [141,142]. This kind of technique tries to avoid the possibility of model overfitting to the specific samples stored in memory. [141] implements a Generative Adversarial Network (GAN) to sample the new instances of data when drifts are detected. However, training both the main model and the GAN could result computationally expensive. One way of addressing this situation without needing a second network is integrating the generative model into the main model by equipping it with generative feedback connections [142].

On the other hand, *Regularization methods* [143–145] impose restrictions in the weight updates to keep them fixed and avoid forgetting an input domain that has already been learnt. [143] propose a state-of-the-art method denominated Elastic Weight Consolidation (EWC), which learns a task, and then determines which connections between weights are determinant to correctly perform that task. Those connections are set applying the matrix of Fisher Information, a statistical tool that weighs the relevance and contribution of each weight to the final result of the model. To do so, it estimates the probability $P(x|y)$, i.e., determines the distribution of inputs as a function of their classes. This approach is usually seen in other works as a baseline method for avoiding catastrophic forgetting. However, it presents some issues, such as scalability and computational efficiency when trying to learn several tasks. [144] introduces a new technique based on EWC which partially solves those issues. Similarly, [145] also consider the conditional probability $P(x|y)$, but in this case they use a Laplacian approximation to reduce the computational costs involved.

Concerning the experimental results, we find the same problem of having very different datasets (see Table 5), making it difficult

Table 5
Summary of the datasets employed in the works presented in Section 5.1. Asterisks indicate that the datasets have been modified in particular ways, making it impossible to fairly compare each other.

Article	Datasets used in experiments	
[140]	MNIST* + CIFAR-10*;	CUBS
[141]	MNIST + SVHN	
[142]	MNIST*	
[143]	MNIST*	
[145]	MNIST* + SVHN + CIFAR-10	

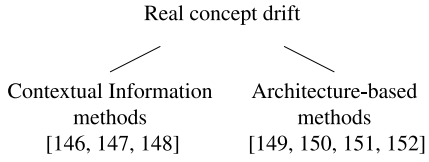


Fig. 6. Classification of the different techniques able to deal with the temporal heterogeneity in the behaviour of the data samples.

to establish relations between the different strategies results. In this case, nonetheless, most of the works we just presented compare their methods with EWC [143], using it as a baseline. In [143], the MNIST dataset is employed to simulate the different input data distributions. They take a random permutation of pixels and apply that permutation to all of the images to create each input domain. With this strategy, they only need one dataset, denoted Permuted MNIST, to simulate any number of domains. In [144], authors compare the accuracy obtained with their learning method (P&C) in each domain, and also the mean accuracy, with EWC, showing that P&C achieves slightly better results avoiding catastrophic forgetting. That also happens in [139], where CLEAR method is compared with both EWC and P&C. CLEAR outperforms EWC in most situations, and attain very similar results to P&C. [140,145] also use the permuted MNIST dataset and improve the mean accuracy of EWC, and some other methods they compare with. Lastly, generative strategies like [141] prove to be efficient to prevent catastrophic forgetting. They use the datasets MNIST and SVHN to show that the performance is barely affected when they change the input dataset.

5.2. Real concept drifts

In this kind of situation, the users are allowed to change the task they are performing during the training process. In Section 3.3.2 we highlighted the necessity of a model designed in a way such that different clients can have distinct outputs even though they own similar inputs. However, our interest now is pursuing a model able to flip from one output to another for the same client in different timestamps. To overcome this challenge, there are two kinds of strategies: On the one hand, we have *Contextual Information Methods*, and on the other hand, we have *Architecture-based Methods*, see Fig. 6.

The approach of *Contextual Information methods* is closely related to the one we talked about in Section 3.3.2 [108]. That work discussed the possibility of adding a piece of information z to the original data inputs, such as a task identifier or a domain identifier. This information z could be designed to address both the multi-domain and the multi-task issues in a variety of situations besides the one presented in the article. Some more research articles in this line are [146–148]. The authors of [146] claim the sought tasks affect the process of training, and propose using a task identifier to achieve personalization. The other approaches [147,148] also rely on some kind of contextual information to determine the task corresponding to each data sample, and on the whole, they act as if they had a specific network for each of the tasks. In this case, when a new task appears in the training stage, each layer of the network is expanded and the new neurons are used for the current task, but not for previous ones.

Table 6
Summary of the datasets employed in the works presented in Section 5.2. Asterisks indicate that the datasets have been modified in particular ways. Some of the datasets mentioned were not referenced so far: Atari games [153].

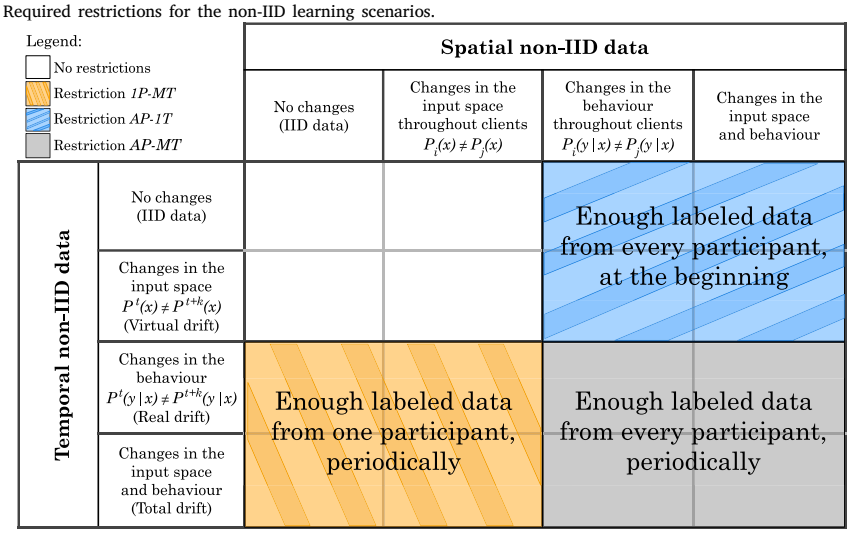
Article	Datasets used in experiments		
[146]	MNIST*;	CIFAR-10*	
[147]	MNIST*;	CIFAR-100	
[148]	MNIST*		
[149]	ImagenNet;	CUBS;	Oxford102Flowers
[150]	ImagenNet;	CUBS;	Oxford102Flowers
[151]	ImagenNet;	CUBS;	Oxford102Flowers
[152]	Atari games		

The other kind of strategies, *Architecture-based methods*, focus on deep incremental multi-task learning techniques that modify the neural network depending on the performing task, without any forgetting on previous tasks. The most studied way for accomplishing it is using some kind of mask on the neural network. Several works have explored this alternative: [149], for instance, study the case of having a neural network already trained for one task and make use of a weight-level binary mask to cancel some of the weights, so the resulting network can solve another previously-defined task. This process can be effectively repeated for several tasks without any forgetting of the original one, as the weights are not effectively changed. There is a slightly different approach proposed by the same authors [150], which consists of starting with a neural network trained for one task, setting some of their weights to zero, i.e, eliminating some neural connections, and retraining the model a few epochs for the initial task. After that, when trying to learn a new task, the already set weights are fixed, and the eliminated connections are reestablished and trained. This method is not scalable to several tasks, as the number of neural connections is limited. Another strategy based on the same ideas uses a ternary neuron-level mask to perform training [151]. The reasoning behind the use of a ternary mask is that some neurons may be useful for both a new task and a previously learned one, so three possible states are considered for each neuron concerning each task: unused, used but not trainable, or trainable. This paper also faces the scalability problem, as they allow the network to grow if necessary, setting the new neurons as unused for previous tasks in order to not modify their accuracy.

On the other hand, authors of [152] propose a completely different technique. They start with a deep neural network for the first task, and when they are interested in learning a new task, they start a new neural network and create connections from the ones that already existed to each layer of the new one, in order to leverage the knowledge from previous tasks. This strategy is really useful when dealing with related tasks, but tasks that interfere with each other might harm the outcome model.

These kinds of methods present a lot of differences in the way they implement their experimental results. Some of them pay attention to the accuracy obtained, while others are more concerned about the error they got, and some others concentrate on the level of forgetting they commit. For instance, in [146] the authors employ the MNIST dataset with pixel permutation, like in [143], and also exchanges some class labels in parts of the dataset to simulate the different behaviours. They compare their results with EWC, LwF [154], and improve their results. However, they emphasize that this form of simulating the different tasks and behaviours is quite unrealistic. Surprisingly, [149–151] employ the same datasets to perform their experiments: the ImageNet dataset [104], used for training the pre-trained ImageNet-VGG-16 neural network; the CUBS dataset [155], and the Oxford102Flowers dataset [156] (see Table 6). This is, as we have seen in this paper, very rare.

Table 7
Required restrictions for the non-IID learning scenarios.



5.3. Data requirements for the different scenarios

To be able to apply federated learning in the different scenarios depicted in Table 4, data has to fulfil certain requirements. In this subsection we will describe these restrictions, which are summarized in Table 7.

Considering all of the scenarios presented in Table 4, some of them are solvable using sophisticated techniques without imposing additional restrictions, but some others may need to verify certain conditions that neither standard FL nor CL demand. As we saw in Sections 3 and 5, facing variations in the marginal input probabilities $P(x)$, either in the spatial or temporal dimension, is possible without any supplementary information [69,74], i.e, unsupervised learning techniques can also be useful in these scenarios. On the contrary, if we seek to detect changes in the conditional probability $P(y|x)$, a certain amount of labelled data is required [108,149], because these kinds of changes can only be measured with the error committed. To be more precise, we establish three restrictions that need to be satisfied to face some scenarios, and denote them as Restrictions 1P-MT, AP-1T, and AP-MT in Table 7:

(i) If the clients behaviour change with time but does not change among the devices, i.e., data fit the cells marked as *Restriction 1P-MT* (one participant, many times) in Table 7, then we will need to have enough labelled data of at least one participant from time to time. Knowing the behaviour of one participant is enough since in these scenarios the behaviours of all of the other participants will be the same. When a Real Concept Drift occurs, that client labelled data will allow the model to detect that drift and properly react to it.

(ii) If, on the contrary, data fit the cells marked as *Restriction AP-1T* (all participants, one time) in Table 7, then the clients would present different conditional probabilities, but they will remain constant in time. In that situation, enough labelled data from all of the participants will be required at the beginning of the training process, so we can determine their initial behaviour. Once their behaviour is settled, it is not possible for it to change, so no more labelled data is required as time passes.

(iii) Lastly, if conditional probabilities vary both in the spatial and temporal axis, which corresponds to cells marked as *Restriction AP-MT* (all participants, many times) in Table 7, then we need enough labelled data from all of the participants, from time to time, so we can conclude when drifts occur and act in consequence. This restriction provides strictly more information than the other ones, so any other scenario considered in Table 7 will also be solvable under this requirement. However, it could be very unrealistic to assume that we could have this information in real-world problems.

The kind of heterogeneity that can be handled without any additional restriction is by far the most studied one in the literature (see the number of works cited in Table 4). Situations where some additional condition is required are less studied, not because the tasks that fit these scenarios are uncommon, but because it is harder to elaborate methods that face this type of issue, as they imply working under the restrictions we just settled.

The spatial and temporal axis can be handled separately since they involve different kinds of techniques, as we have previously seen, and they are independent sources of heterogeneity. A realistic problem may present both of them, and hence the model developed to solve it must implement some tool that addresses each of the possible kinds of heterogeneity.

Tools to deal with spatial and temporal heterogeneity are respectively orthogonal, i.e., they do not affect nor interfere on each other. For instance, the usage of time windows to detect any kind of drift, and a domain factorization strategy on each client to adapt the different feature domains, are two strategies that can be deployed at the same time and have no impact on each other. Therefore, when dealing with a situation that presents spatial and temporal non-IID data at the same time, each source of heterogeneity can be addressed independently with the suitable information, i.e., the restrictions we mentioned.

6. Experiments

In order to solidly establish that the restrictions presented in Section 5.3 are essential, and that the corresponding heterogeneity could not be handled without them, in this section we present several experiments. We aim to illustrate how the different types of non-IID data deteriorate the models obtained and decreases their performances. We divide our experiments in two groups, the ones that present spatial non-IID data (Section 6.1) and those that present temporal non-IID data (Section 6.2).

For all of the experiments we employed the *Digit-five* dataset, which includes MNIST, MNIST-M, SVHN, USPS and Synthetic [79], all gathered. We employ this one because the input images present lots of different aspects, and hence each of the datasets can represent a different domain. This way we avoid making our own modifications into $P(x)$ to get non-IID data, both among the participants and over time. We restricted the data in each of the 5 domains to 60,000 samples,

so we have a total of 300,000 patterns. We distributed them across 50 clients, so each of them owns a total of 6000 data samples.

We performed experiments in two different scenarios, one that presents spatial non-IID data and one that presents temporal non-IID data. In each of them, some particularities about the problem setting and the data processing must differ to properly represent each situation, so further details are explained in Sections 6.1 and 6.2. The model architecture employed was the same in all of the experiments, and consists of a simple Convolutional Neural Network (CNN) with 4 convolutional layers followed by 3 dense layers. In addition, we have ran each experiment multiple times to make sure the results were statistically significant and no artefacts had been produced.

6.1. Spatial non-IID scenarios

In this scenario, data varies across clients, but remains the same along time. To achieve this kind of heterogeneity, we present 4 different realistic cases that help to understand how the data distributions across clients affects the performance of some FL models. For our experiments we selected two different algorithms, *FedAvg* and *FedProx*. Recall that *FedProx* [52] is a method designed to deal with changes in $P(x)$ across clients. In all of the experiments, we selected 35 clients for the training process, and the data from the rest of them (15 clients) was employed to perform the testing of the models obtained.

Scenario A. Spatial baseline. The first scenario we present is the baseline. In this scenario, each client owns 1200 samples from each of the 5 domains, so the situation is totally IID, which may seem unrealistic. As illustrated in 7(a) and 7(b), in this scenario both *FedAvg* and *FedProx* perform excellently well, reaching an accuracy of 90%.

Scenario B. Input space heterogeneity with homogeneous test. In this scenario, the dataset of each client only has data from 1 domain, and the 15 clients selected for testing are randomly selected. In 7(c) and 7(d) we can see that the global model from *FedAvg* and *FedProx* achieve a performance similar to the baseline scenario (A), although the situation presented here is more realistic, since each client presents its own kind of data. Both methods take a couple more training rounds to converge, but the difference is not significant, meaning that both *FedAvg* and *FedProx* can deal with this kind of non-IID data.

Scenario C. Input space heterogeneity with biased test. This scenario is very similar to Scenario *B*, with the only difference that all of the clients that have MNIST data samples are selected for testing, and for that reason, there are no MNIST data samples in the training clients datasets. This emulates the situation presented in the *Domain Adaptation* techniques (See Section 3.3.1), i.e., a new domain appears in testing time. In 7(e) and 7(f) we can see that the accuracy drops significantly in this scenario for both *FedAvg* and *FedProx*, which is very reasonable because most of the data used for testing the model belongs to a domain what was unseen during training. At the same time, *FedProx* performs significantly better than *FedAvg* (68% and 48% respectively), because it is prepared to face this kind of heterogeneity.

Scenario D. Behaviour heterogeneity. This last scenario emulates a situation where some of the participants present different behaviours. As in Scenario B, each client only has data samples from 1 domain, and the clients selected for testing are randomized. However, we have altered all of the labels from the SVHN dataset, so 10 of the clients present data labelled differently with respect to the others, and 3 of them are selected for testing. In 7(g) and 7(h) we can see that *FedAvg* and *FedProx* obtain similar accuracies, 63% and 64% respectively, because neither of them is designed to face this kind of heterogeneity.

6.2. Temporal non-IID scenarios

In this scenario, data varies along time, but it presents the same characteristics and properties in all of the clients, and it varies at the same time for all of them. This type of temporal evolution is much predictable and affordable than what we would expect in a real-life problem, but it is enough to deteriorate the federated models, as we illustrate ahead. Again, for our experiments we selected two different algorithms, *FedAvg* and *CDA-FedAvg*. Recall that *CDA-FedAvg* [8] is a method designed to face changes in $P(x)$ along time. On the other hand, *FedAvg* is not a method designed for CL. It expect to have all the data available from the beginning of the training process, and that is not the case here.

To naively adapt *FedAvg* to this setting, we have fixed that each client selected for training processes a significant quantity of data samples from their dataset, 800, and use them to perform the training stage. After that, those data samples are stored and the next time that client trains a model will be using those 800 samples as well as new 800 data samples more. We have also fixed a maximum amount of data, 5000, that each client can store and use for training, i.e., after that number of samples is processed, the training dataset will start forgetting the first samples collected. The purpose of implementing this restriction is trying to emulate a real problem, in which the different devices may not be capable of handling all the information they capture. This same quantities and limitations are fixed for *CDA-FedAvg*, although this method only stores data samples when detecting a drift, so its management of resources is more efficient.

Scenario A. Temporal baseline. The first scenario we examine is the baseline. In this scenario, each client owns 1200 samples from each of the 5 domains, and they are shuffled so the domains are all mixed together. This situation is totally IID, since each client is going to perceive, in each training round, 800 samples that represent the 5 domains. As illustrated in 8(a) and 8(b), in this scenario both *FedAvg* and *CDA-FedAvg* perform well, reaching an accuracy of 84% in both cases. They perform worse that in the Spatial baseline, Figs. 7(a) and 7(b), because of the way they are processing the data and the limitation of 5000 data samples.

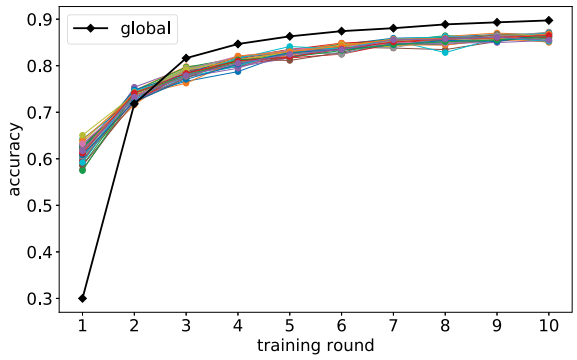
Scenario B. Virtual drift. This other scenario presents virtual drifts in all of the clients simultaneously. Each of them has 1200 samples from each domain, but in this case they are not shuffled, i.e., the first 1200 samples are from the MNIST dataset, the next 1200 are from the SVHN dataset, and so on. The results in this case show that *FedAvg* is seriously affected by this kind of data heterogeneity (see Fig. 8(c)), lowering its accuracy to 48%. On the other hand, *CDA-FedAvg* can handle this situation (see Fig. 8(d)), achieving an accuracy of 79%.

7. Challenges and future directions

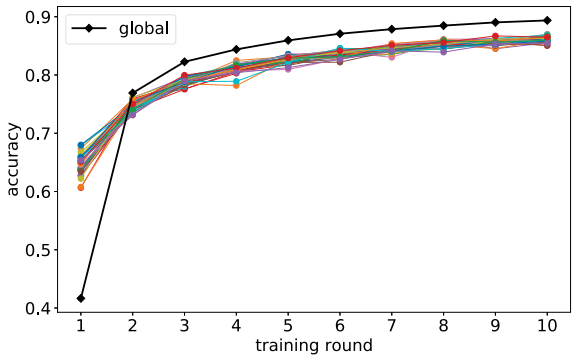
Along this review, we have discussed the different possible causes of data heterogeneity, as well as the most common and remarkable strategies developed so far to face it. Some of those strategies are already designed and implemented in the FL framework, while some others seem promising, but at the moment are only conceived in centralized settings. At the same time, we have addressed the necessity of considering time-evolving methods for real-life federated problems. Some works are aware of this kind of issue, but nowadays this area of research is much less studied than the one regarding non-IID data.

Concerning the non-IID data, there are some important existing problems. One of the biggest ones is that most of the strategies designed to tackle non-IID data do not specify what kind of non-IID data source they work with. See, for instance, works presented in Sections 3.2 and 3.3. Hence, when trying to apply some method for a real-life problem, it is unclear to determine which ones are useful, or if some of them are more appropriate than others. Also, the fact that a lots of works claim to deal with non-IID data leads to the thinking that there are a lot of different techniques in the current literature to solve non-IID data problems, but the reality is that some kinds of heterogeneity are still barely studied.

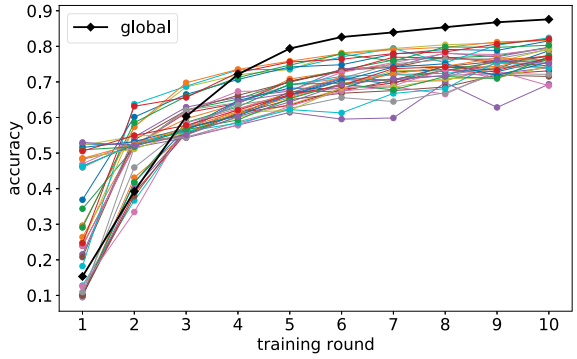
Nowadays, personalization strategies (Section 3.2) are gaining a lot of importance in FL. These methods are aware of the possibility of having clients that need their own outputs for their data. On the contrary, most of the current strategies in the literature assume that



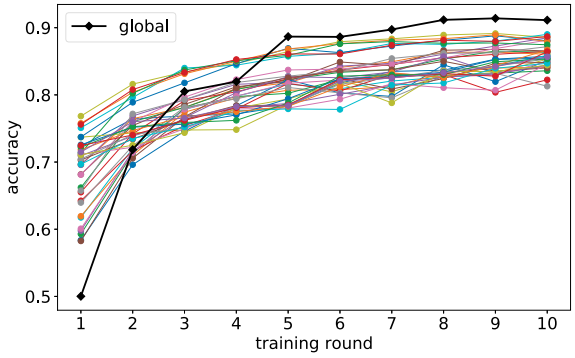
(a) Spatial baseline – FedAvg.



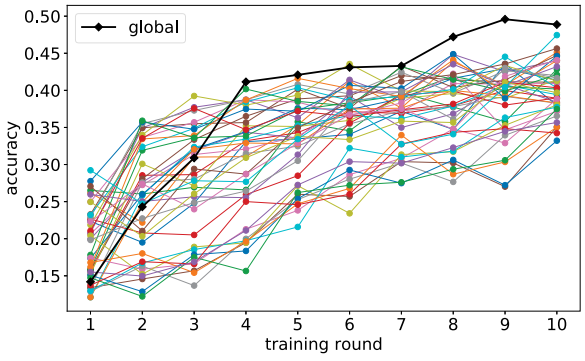
(b) Spatial baseline – FedProx.



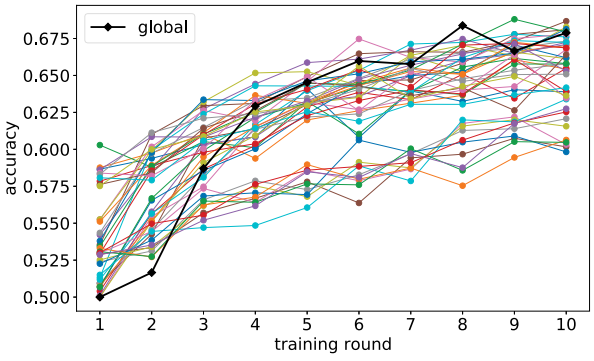
(c) Input space heterogeneity with homogeneous test – FedAvg.



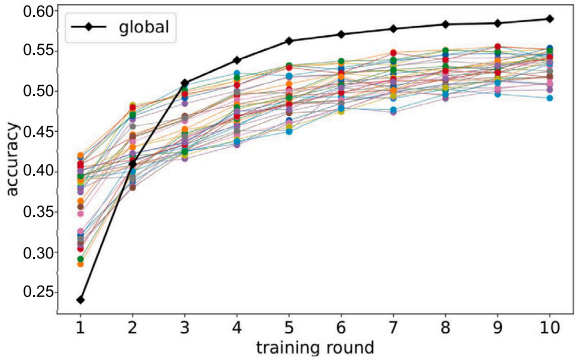
(d) Input space heterogeneity with homogeneous test – FedProx.



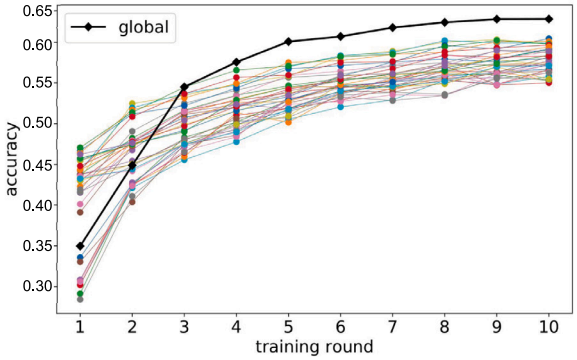
(e) Input space heterogeneity with biased test – FedAvg.



(f) Input space heterogeneity with biased test – FedProx.

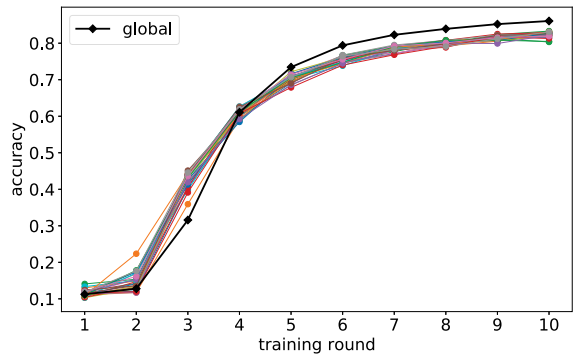


(g) Behaviour heterogeneity – FedAvg.

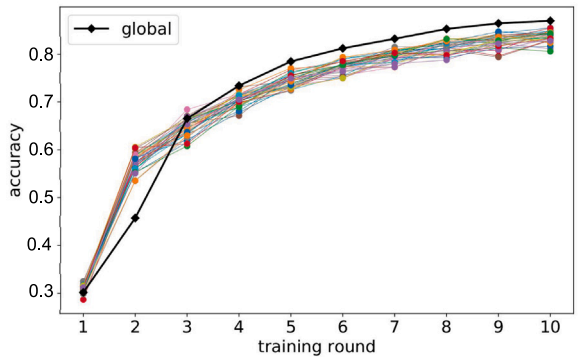


(h) Behaviour heterogeneity – FedProx.

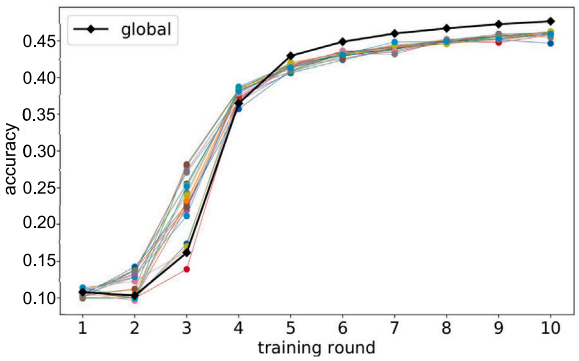
Fig. 7. Performances of FedAvg and FedProx in the different Spatial non-IID Scenarios. The black thick line represents the global model accuracy, whereas the other ones represent the accuracy achieved by the model of each client.



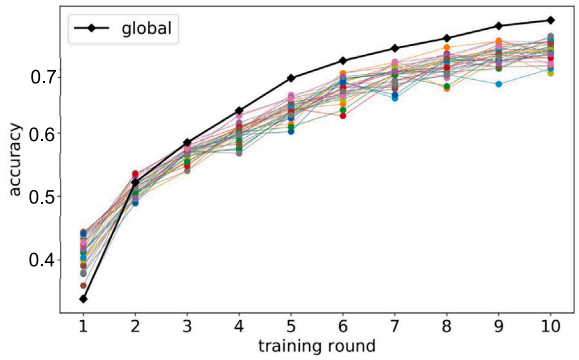
(a) Temporal baseline – FedAvg.



(b) Temporal baseline – CDA-FedAvg.



(c) Virtual drift – FedAvg.



(d) Virtual drift – CDA-FedAvg.

Fig. 8. Performances of FedAvg and CDA-FedAvg in the different Temporal non-IID Scenarios. The black thick line represents the global model accuracy, whereas the other ones represent the accuracy of the model of each client.

the same input data belonging to different participants must be replied to with the same output. As we saw in Sections 3.3.2 and 5.2, this is not always true.

Moreover, some of the proposed techniques for handling spatial non-IID data are already conceived in a FL framework, such as Generative Adversarial Networks. Nonetheless, some strategies are yet to be deployed in FL settings. Is the case of Domain Factorization methods and Dissimilarity methods. Regarding Domain Factorization, the main challenge when trying to perform these strategies in a federated setting is that each participant would construct a different input space factorization, and it is necessary to establish a common ground for all of them. Concerning Dissimilarity methods, the main challenge is establishing a common metric that generalizes the domain variability of all participants.

Concerning the temporal dimension, it is important to notice that the current strategies of drift detection and adaptation are mostly deployed in centralized settings. However, we selected the strategies they employ because they could be easily adapted to federated settings. For instance, in a federated environment, each of the participants could perform a rehearsal technique to avoid forgetting previous concepts. Another possibility, considering Regularization Methods, is that clients who perceive different domains train particular neurons of the network, leading to a faster domain adaptation. Nonetheless, some difficulties can arise in these situations:

- Clients may experience drifts at different timestamps, and thus they present different input domains simultaneously. This can lead to very different updates for the global model, and prevent the global model from converging.
- Clients may experience similar drifts in their data without being aware of it, and mechanisms that provide this information would facilitate achieving a better model faster.

In addition to the inherent difficulties of considering heterogeneous data, the lack of specific datasets makes it harder to test the quality of the methods under these settings. At the present moment, each work employs different datasets to test their methods, and in the majority of cases, they need to modify those datasets to create the desired data distributions (see Tables 2, 3, 5 and 6). Under these circumstances, it is impossible to fairly compare those methods. It is necessary to have a common benchmark dataset, with standard representations of some types of heterogeneous data. This would allow to contrast the current and future strategies over a common set of data and properly compare them.

CRediT authorship contribution statement

Marcos F. Criado: Formal analysis, Investigation, Conceptualization, Software, Visualization, Data presentation, Validation, Writing – original draft. **Fernando E. Casado:** Formal analysis, Investigation, Conceptualization, Software, Writing – review & editing, Visualization, Data presentation. **Roberto Iglesias:** Supervision, Funding acquisition. **Carlos V. Regueiro:** Supervision, Funding acquisition. **Senén Barro:** Supervision, Funding acquisition, Project administration.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgements

This work has received financial support from AEI/FEDER (EU) grant number PID2020-119367RB-I00. It has also been supported by the Xunta de Galicia - Consellería de Cultura, Educación e Universidade (Centros de investigación de Galicia accreditation 2019–2022 ED431G-2019/04 and ED431G2019/01, and Reference Competitive Groups accreditation 2021–2024, ED431C 2018/29, ED431F2018/02 and ED431C 2021/30) and the European Union (European Regional Development Fund - ERDF). Finally, it has also been funded by the Ministerio de Universidades of Spain in the FPU 2017 program (FPU17/04154).

References

[1] J. Konečnỳ, B. McMahan, D. Ramage, Federated optimization: Distributed optimization beyond the datacenter, 2015, arXiv preprint arXiv:1511.03575.
[2] B. McMahan, E. Moore, D. Ramage, S. Hampson, B.A. y Arcas, Communication-efficient learning of deep networks from decentralized data, in: Artificial Intelligence and Statistics, PMLR, 2017, pp. 1273–1282.
[3] W.Y.B. Lim, N.C. Luong, D.T. Hoang, Y. Jiao, Y.-C. Liang, Q. Yang, D. Niyato, C. Miao, Federated learning in mobile edge networks: A comprehensive survey, IEEE Commun. Surv. Tutor. 22 (3) (2020) 2031–2063.
[4] T. Li, A.K. Sahu, A. Talwalkar, V. Smith, Federated learning: Challenges, methods, and future directions, IEEE Signal Process. Mag. 37 (3) (2020) 50–60.
[5] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, V. Chandra, Federated learning with non-iid data, 2018, arXiv preprint arXiv:1806.00582.
[6] G.I. Parisi, R. Kemker, J.L. Part, C. Kanan, S. Wermter, Continual lifelong learning with neural networks: A review, Neural Netw. 113 (2019) 54–71.
[7] J. Gama, I. Žliobaitė, A. Bifet, M. Pechenizkiy, A. Bouchachia, A survey on concept drift adaptation, ACM Comput. Surv. 46 (4) (2014) 1–37.
[8] F.E. Casado, D. Lema, M.F. Criado, R. Iglesias, C.V. Regueiro, S. Barro, Concept drift detection and adaptation for federated and continual learning, Multimedia Tools Appl. (2021).
[9] A. Usmanova, F. Portet, P. Lalanda, G. Vega, A distillation-based approach integrating continual learning and federated learning for pervasive services, 2021, arXiv preprint arXiv:2109.04197.
[10] T.J. Park, K. Kumatani, D. Dimitriadis, Tackling dynamics in federated incremental learning with variational embedding rehearsal, 2021, arXiv preprint arXiv:2110.09695.
[11] H.B. McMahan, E. Moore, D. Ramage, B. Aguera-Arcas, Federated learning of deep networks using model averaging, 2016, arXiv Preprint arXiv:1602.05629v1.
[12] M. Servetnyk, C.C. Fung, Z. Han, Unsupervised federated learning for unbalanced data, in: GLOBECOM 2020-2020 IEEE Global Communications Conference, IEEE, 2020, pp. 1–6.
[13] E. Tzinis, J. Casebeer, Z. Wang, P. Smaragdis, Separate but together: Unsupervised federated learning for speech enhancement from non-IID data, in: 2021 IEEE Workshop on Applications of Signal Processing to Audio and Acoustics, WASPAA, IEEE, 2021, pp. 46–50.
[14] B. Custers, A.M. Sears, F. Dechesne, I. Georgieva, T. Tani, S. van der Hof, EU Personal Data Protection in Policy and Practice, Springer, 2019.
[15] B.M. Gaff, H.E. Sussman, J. Geetter, Privacy and big data, Computer 47 (6) (2014) 7–9.
[16] L. Lyu, H. Yu, Q. Yang, Threats to federated learning: A survey, 2020, arXiv preprint arXiv:2003.02133.
[17] S. Truex, N. Baracaldo, A. Anwar, T. Steinke, H. Ludwig, R. Zhang, Y. Zhou, A hybrid approach to privacy-preserving federated learning, in: Proceedings of the 12th ACM Workshop on Artificial Intelligence and Security, 2019, pp. 1–11.
[18] Z. Wang, M. Song, Z. Zhang, Y. Song, Q. Wang, H. Qi, Beyond inferring class representatives: User-level privacy leakage from federated learning, in: IEEE INFOCOM 2019-IEEE Conference on Computer Communications, IEEE, 2019, pp. 2512–2520.
[19] M. Naehrig, K. Lauter, V. Vaikuntanathan, Can homomorphic encryption be practical? in: Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop, 2011, pp. 113–124.
[20] L.J. Aslett, P.M. Esperança, C.C. Holmes, A review of homomorphic encryption and software tools for encrypted statistical machine learning, Stat 1050 (2015) 26.
[21] K. Wei, J. Li, M. Ding, C. Ma, H.H. Yang, F. Farokhi, S. Jin, T.Q. Quek, H.V. Poor, Federated learning with differential privacy: Algorithms and performance analysis, IEEE Trans. Inf. Forensics Secur. 15 (2020) 3454–3469.
[22] R.C. Geyer, T. Klein, M. Nabi, Differentially private federated learning: A client level perspective, 2017, arXiv preprint arXiv:1712.07557.
[23] C. Zhao, S. Zhao, M. Zhao, Z. Chen, C.-Z. Gao, H. Li, Y.-a. Tan, Secure multi-party computation: theory, practice and applications, Inform. Sci. 476 (2019) 357–372.
[24] Y. Liu, Y. Kang, C. Xing, T. Chen, Q. Yang, A secure federated transfer learning framework, IEEE Intell. Syst. 35 (4) (2020) 70–82.
[25] F.E. Casado, D. Lema, R. Iglesias, C.V. Regueiro, S. Barro, Federated and continual learning for classification tasks in a society of devices, 2020, arXiv preprint arXiv:2006.07129.
[26] Y. Chen, Y. Ning, M. Slawski, H. Rangwala, Asynchronous online federated learning for edge devices with non-iid data, in: 2020 IEEE International Conference on Big Data (Big Data), IEEE, 2020, pp. 15–24.
[27] X. Li, Z. Qu, B. Tang, Z. Lu, Stragglers are not disaster: A hybrid federated learning algorithm with delayed gradients, 2021, arXiv preprint arXiv:2102.06329.
[28] N. Rodríguez-Barroso, E. Martínez-Cámara, M. Luzón, G.G. Seco, M.A. Veganzones, F. Herrera, Dynamic federated learning model for identifying adversarial clients, 2020, arXiv preprint arXiv:2007.15030.
[29] S. Wang, T. Tuor, T. Salonidis, K.K. Leung, C. Makaya, T. He, K. Chan, Adaptive federated learning in resource constrained edge computing systems, IEEE J. Sel. Areas Commun. 37 (6) (2019) 1205–1221.
[30] C.T. Dinh, N.H. Tran, M.N. Nguyen, C.S. Hong, W. Bao, A.Y. Zomaya, V. Gramoli, Federated learning over wireless networks: Convergence analysis and resource allocation, IEEE/ACM Trans. Netw. (2020).
[31] P. Kairouz, H.B. McMahan, B. Avent, A. Bellet, M. Bennis, A.N. Bhagoji, K. Bonawitz, Z. Charles, G. Cormode, R. Cummings, et al., Advances and open problems in federated learning, 2019, arXiv preprint arXiv:1912.04977.
[32] S. Caldas, S.M.K. Duddu, P. Wu, T. Li, J. Konečnỳ, H.B. McMahan, V. Smith, A. Talwalkar, Leaf: A benchmark for federated settings, 2018, arXiv preprint arXiv:1812.01097.
[33] Y. LeCun, L. Bottou, Y. Bengio, P. Haffner, Gradient-based learning applied to document recognition, Proc. IEEE 86 (11) (1998) 2278–2324.
[34] Y. Ganin, E. Ustinova, H. Ajakan, P. Germain, H. Larochelle, F. Laviolette, M. Marchand, V. Lempitsky, Domain-adversarial training of neural networks, J. Mach. Learn. Res. 17 (1) (2016) 2030–2096.
[35] N. Mu, J. Gilmer, Mnist-c: A robustness benchmark for computer vision, 2019, arXiv preprint arXiv:1906.02337.
[36] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-iid data with reinforcement learning, in: IEEE INFOCOM 2020-IEEE Conference on Computer Communications, IEEE, 2020, pp. 1698–1707.
[37] F. Sattler, S. Wiedemann, K.-R. Müller, W. Samek, Robust and communication-efficient federated learning from non-iid data, IEEE Trans. Neural Netw. Learn. Syst. 31 (9) (2019) 3400–3413.
[38] X. Li, K. Huang, W. Yang, S. Wang, Z. Zhang, On the convergence of fedavg on non-iid data, 2019, arXiv preprint arXiv:1907.02189.
[39] Q. Li, Y. Diao, Q. Chen, B. He, Federated learning on non-iid data silos: An experimental study, 2021, arXiv preprint arXiv:2102.02079.
[40] D. Li, J. Wang, Fedmd: Heterogenous federated learning via model distillation, 2019, arXiv preprint arXiv:1910.03581.
[41] K. Wang, R. Mathews, C. Kiddon, H. Eichner, F. Beaufays, D. Ramage, Federated evaluation of on-device personalization, 2019, arXiv preprint arXiv:1910.10252.
[42] P.P. Liang, T. Liu, L. Ziyin, N.B. Allen, R.P. Auerbach, D. Brent, R. Salakhutdinov, L.-P. Morency, Think locally, act globally: Federated learning with local and global representations, 2020, arXiv preprint arXiv:2001.01523.
[43] C. T. Dinh, N. Tran, J. Nguyen, Personalized federated learning with moreau envelopes, Adv. Neural Inf. Process. Syst. 33 (2020) 21394–21405.
[44] Z. Ma, Y. Lu, W. Li, J. Yi, S. Cui, PFedAtt: Attention-based personalized federated learning on heterogeneous clients, in: Asian Conference on Machine Learning, PMLR, 2021, pp. 1253–1268.
[45] C. Finn, P. Abbeel, S. Levine, Model-agnostic meta-learning for fast adaptation of deep networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 1126–1135.
[46] A. Fallah, A. Mokhtari, A. Ozdaglar, Personalized federated learning: A meta-learning approach, 2020, arXiv preprint arXiv:2002.07948.
[47] T. Yu, T. Li, Y. Sun, S. Nanda, V. Smith, V. Sekar, S. Seshan, Learning context-aware policies from multiple smart homes via federated multi-task learning, in: 2020 IEEE/ACM Fifth International Conference on Internet-of-Things Design and Implementation, IoTDI, IEEE, 2020, pp. 104–115.
[48] V. Smith, C.-K. Chiang, M. Sanjabi, A. Talwalkar, Federated multi-task learning, 2017, arXiv preprint arXiv:1705.10467.
[49] M.G. Arivazhagan, V. Aggarwal, A.K. Singh, S. Choudhary, Federated learning with personalization layers, 2019, arXiv preprint arXiv:1912.00818.
[50] H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, pp. 4293–4302.
[51] F. Hanzely, P. Richtárik, Federated learning of a mixture of global and local models, 2020, arXiv preprint arXiv:2002.05516.
[52] T. Li, A.K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, V. Smith, Federated optimization in heterogeneous networks, 2018, arXiv preprint arXiv:1812.06127.
[53] Y. Deng, M.M. Kamani, M. Mahdavi, Adaptive personalized federated learning, 2020, arXiv preprint arXiv:2003.13461.
[54] N. Tajbakhsh, J.Y. Shin, S.R. Gurudu, R.T. Hurst, C.B. Kendall, M.B. Gotway, J. Liang, Convolutional neural networks for medical image analysis: Full training or fine tuning? IEEE Trans. Med. Imaging 35 (5) (2016) 1299–1312.

[55] S.A. Dudani, The distance-weighted k-nearest-neighbor rule, IEEE Trans. Syst. Man Cybern. (1976) 325–327.

[56] J. Ren, X. Shen, Z. Lin, R. Mech, D.J. Foran, Personalized image aesthetics, in: Proceedings of the IEEE International Conference on Computer Vision, 2017, pp. 638–647.

[57] A. Krizhevsky, G. Hinton, et al., Learning multiple layers of features from tiny images, 2009.

[58] F. Sattler, K.-R. Müller, W. Samek, Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints, IEEE Trans. Neural Netw. Learn. Syst. (2020).

[59] N. Shlezinger, S. Rini, Y.C. Eldar, The communication-aware clustered federated learning problem, in: 2020 IEEE International Symposium on Information Theory, ISIT, IEEE, 2020, pp. 2610–2615.

[60] C. Briggs, Z. Fan, P. Andras, Federated learning with hierarchical clustering of local updates to improve training on non-IID data, in: 2020 International Joint Conference on Neural Networks, IJCNN, IEEE, 2020, pp. 1–9.

[61] J. Hoffman, M. Mohri, N. Zhang, Algorithms and theory for multiple-source adaptation, 2018, arXiv preprint arXiv:1805.08727.

[62] M. Mohri, G. Sivek, A.T. Suresh, Agnostic federated learning, in: International Conference on Machine Learning, PMLR, 2019, pp. 4615–4625.

[63] Y. Mansour, M. Mohri, J. Ro, A.T. Suresh, Three approaches for personalization with applications to federated learning, 2020, arXiv preprint arXiv:2002.10619.

[64] G. Cohen, S. Afshar, J. Tapson, A. Van Schaik, EMNIST: Extending MNIST to handwritten letters, in: 2017 International Joint Conference on Neural Networks, IJCNN, IEEE, 2017, pp. 2921–2926.

[65] B. Lake, R. Salakhutdinov, J. Gross, J. Tenenbaum, One shot learning of simple visual concepts, in: Proceedings of the Annual Meeting of the Cognitive Science Society, 33, 2011.

[66] Y. Wu, J. Lim, M.-H. Yang, Online object tracking: A benchmark, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2013, pp. 2411–2418.

[67] S. Hadfield, K. Lebeda, R. Bowden, The visual object tracking VOT2014 challenge results, in: European Conference on Computer Vision (ECCV) Visual Object Tracking Challenge Workshop, University of Surrey, 2014.

[68] H. Xiao, K. Rasul, R. Vollgraf, Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms, 2017, arXiv preprint arXiv:1708.07747.

[69] H. Zhao, R.T. Des Combes, K. Zhang, G. Gordon, On learning invariant representations for domain adaptation, in: International Conference on Machine Learning, PMLR, 2019, pp. 7523–7532.

[70] A.-A. Liu, N. Xu, W.-Z. Nie, Y.-T. Su, Y.-D. Zhang, Multi-domain and multi-task learning for human action recognition, IEEE Trans. Image Process. 28 (2) (2018) 853–867.

[71] J. Hoffman, B. Kulis, T. Darrell, K. Saenko, Discovering latent domains for multisource domain adaptation, in: European Conference on Computer Vision, Springer, 2012, pp. 702–715.

[72] F. Siyahjani, R. Almohsen, S. Sabri, G. Doretto, A supervised low-rank method for learning invariant subspaces, in: Proceedings of the IEEE International Conference on Computer Vision, 2015, pp. 4220–4228.

[73] C. Zhang, H. Zhang, L.E. Parker, Feature space decomposition for effective robot adaptation, in: 2015 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS, IEEE, 2015, pp. 441–448.

[74] M. Wang, B. Liu, J. Tang, X.-S. Hua, Metric learning with feature decomposition for image categorization, Neurocomputing 73 (10–12) (2010) 1562–1569.

[75] K.Q. Weinberger, L.K. Saul, Distance metric learning for large margin nearest neighbor classification, J. Mach. Learn. Res. 10 (2) (2009).

[76] L. Yang, R. Jin, Distance metric learning: A comprehensive survey, Michigan State Univ. 2 (2) (2006) 4.

[77] E. Xing, M. Jordan, S.J. Russell, A. Ng, Distance metric learning with application to clustering with side-information, Adv. Neural Inf. Process. Syst. 15 (2002) 521–528.

[78] I.H. Daumé, Frustratingly easy domain adaptation, 2009, arXiv preprint arXiv:0907.1815.

[79] Y. Ganin, V. Lempitsky, Unsupervised domain adaptation by backpropagation, in: International Conference on Machine Learning, PMLR, 2015, pp. 1180–1189.

[80] K. You, M. Long, Z. Cao, J. Wang, M.I. Jordan, Universal domain adaptation, in: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, 2019, pp. 2720–2729.

[81] Y. Chen, X. Qin, J. Wang, C. Yu, W. Gao, Fedhealth: A federated transfer learning framework for wearable healthcare, IEEE Intell. Syst. 35 (4) (2020) 83–93.

[82] M. Long, H. Zhu, J. Wang, M.I. Jordan, Deep transfer learning with joint adaptation networks, in: International Conference on Machine Learning, PMLR, 2017, pp. 2208–2217.

[83] X. Peng, Q. Bai, X. Xia, Z. Huang, K. Saenko, B. Wang, Moment matching for multi-source domain adaptation, in: Proceedings of the IEEE/CVF International Conference on Computer Vision, 2019, pp. 1406–1415.

[84] P.O. Pinheiro, Unsupervised domain adaptation with similarity learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 8004–8013.

[85] M. Baktashmotlagh, M.T. Harandi, B.C. Lovell, M. Salzmann, Unsupervised domain adaptation by domain invariant projection, in: Proceedings of the IEEE International Conference on Computer Vision, 2013, pp. 769–776.

[86] M. Dredze, K. Crammer, Online methods for multi-domain learning and adaptation, in: Proceedings of the 2008 Conference on Empirical Methods in Natural Language Processing, 2008, pp. 689–697.

[87] M. Dredze, A. Kulesza, K. Crammer, Multi-domain learning by confidence-weighted parameter combination, Mach. Learn. 79 (1–2) (2010) 123–149.

[88] T. Van Erven, P. Harremos, Rényi divergence and Kullback-Leibler divergence, IEEE Trans. Inform. Theory 60 (7) (2014) 3797–3820.

[89] E. Tzeng, J. Hoffman, K. Saenko, T. Darrell, Adversarial discriminative domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 7167–7176.

[90] Z. Pei, Z. Cao, M. Long, J. Wang, Multi-adversarial domain adaptation, in: Thirty-Second AAAI Conference on Artificial Intelligence, 2018.

[91] Z. Cao, L. Ma, M. Long, J. Wang, Partial adversarial domain adaptation, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 135–150.

[92] S. Motiian, Q. Jones, S.M. Iranmanesh, G. Doretto, Few-shot adversarial domain adaptation, 2017, arXiv preprint arXiv:1711.02536.

[93] X. Peng, Z. Huang, Y. Zhu, K. Saenko, Federated adversarial domain adaptation, 2019, arXiv preprint arXiv:1911.02054.

[94] M. Long, Y. Cao, J. Wang, M. Jordan, Learning transferable features with deep adaptation networks, in: International Conference on Machine Learning, PMLR, 2015, pp. 97–105.

[95] E. Tzeng, J. Hoffman, N. Zhang, K. Saenko, T. Darrell, Deep domain confusion: Maximizing for domain invariance, 2014, arXiv preprint arXiv:1412.3474.

[96] M. Long, H. Zhu, J. Wang, M.I. Jordan, Unsupervised domain adaptation with residual transfer networks, 2016, arXiv preprint arXiv:1602.04433.

[97] K. Saenko, B. Kulis, M. Fritz, T. Darrell, Adapting visual category models to new domains, in: European Conference on Computer Vision, Springer, 2010, pp. 213–226.

[98] H. Venkateswara, J. Eusebio, S. Chakraborty, S. Panchanathan, Deep hashing network for unsupervised domain adaptation, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 5018–5027.

[99] X. Peng, B. Usman, N. Kaushik, J. Hoffman, D. Wang, K. Saenko, Visda: The visual domain adaptation challenge, 2017, arXiv preprint arXiv:1710.06924.

[100] Y. Netzer, T. Wang, A. Coates, A. Bissacco, B. Wu, A.Y. Ng, Reading digits in natural images with unsupervised feature learning, 2011.

[101] R. Gopalan, R. Li, R. Chellappa, Domain adaptation for object recognition: An unsupervised approach, in: 2011 International Conference on Computer Vision, IEEE, 2011, pp. 999–1006.

[102] A. Bergamo, L. Torresani, Exploiting weakly-labeled web images to improve object classification: a domain adaptation approach, Adv. Neural Inf. Process. Syst. 23 (2010) 181–189.

[103] Y. Chen, J. Bi, J.Z. Wang, MILES: Multiple-instance learning via embedded instance selection, IEEE Trans. Pattern Anal. Mach. Intell. 28 (12) (2006) 1931–1947.

[104] O. Russakovsky, J. Deng, H. Su, J. Krause, S. Satheesh, S. Ma, Z. Huang, A. Karpathy, A. Khosla, M. Bernstein, et al., Imagenet large scale visual recognition challenge, Int. J. Comput. Vis. 115 (3) (2015) 211–252.

[105] N. Silberman, D. Hoiem, P. Kohli, R. Fergus, Indoor segmentation and support inference from rgbd images, in: European Conference on Computer Vision, Springer, 2012, pp. 746–760.

[106] X.-D. Zhang, Machine learning, in: A Matrix Algebra Approach to Artificial Intelligence, Springer, 2020, pp. 223–440.

[107] T. Hiessl, Cohort-based federated learning services for industrial collaboration on the edge, 2021.

[108] Y. Yang, T.M. Hospedales, A unified perspective on multi-domain and multi-task learning, 2014, arXiv preprint arXiv:1412.7489.

[109] L. Corinzia, A. Beuret, J.M. Buhmann, Variational federated multi-task learning, 2019, arXiv preprint arXiv:1906.06268.

[110] T. Lesort, V. Lomonaco, A. Stoian, D. Maltoni, D. Filliat, N. Díaz-Rodríguez, Continual learning for robotics: Definition, framework, learning strategies, opportunities and challenges, Inf. Fusion 58 (2020) 52–68.

[111] S. Thrun, T.M. Mitchell, Lifelong robot learning, Robot. Auton. Syst. 15 (1–2) (1995) 25–46.

[112] C. Tessler, S. Givony, T. Zahavy, D. Mankowitz, S. Mannor, A deep hierarchical approach to lifelong learning in minecraft, in: Proceedings of the AAAI Conference on Artificial Intelligence, 31, 2017.

[113] A. Carlson, J. Betteridge, B. Kisiel, B. Settles, E.R. Hruschka, T.M. Mitchell, Toward an architecture for never-ending language learning, in: Twenty-Fourth AAAI Conference on Artificial Intelligence, 2010.

[114] T. Mitchell, W. Cohen, E. Hruschka, P. Talukdar, B. Yang, J. Betteridge, A. Carlson, B. Dalvi, M. Gardner, B. Kisiel, et al., Never-ending learning, Commun. ACM 61 (5) (2018) 103–115.

[115] T. Xiao, J. Zhang, K. Yang, Y. Peng, Z. Zhang, Error-driven incremental learning in deep convolutional neural network for large-scale image classification, in: Proceedings of the 22nd ACM International Conference on Multimedia, 2014, pp. 177–186.

[116] A. Gepperth, B. Hammer, Incremental learning algorithms and applications, in: European Symposium on Artificial Neural Networks, ESANN, 2016.

[117] S.-A. Rebuffi, A. Kolesnikov, G. Sperl, C.H. Lampert, Icarl: Incremental classifier and representation learning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2017, pp. 2001–2010.

[118] J. Yoon, W. Jeong, G. Lee, E. Yang, S.J. Hwang, Federated continual learning with weighted inter-client transfer, in: International Conference on Machine Learning, PMLR, 2021, pp. 12073–12086.

[119] R. Kemker, M. McClure, A. Abitino, T. Hayes, C. Kanan, Measuring catastrophic forgetting in neural networks, in: Proceedings of the AAAI Conference on Artificial Intelligence, vol. 32, 2018.

[120] I.J. Goodfellow, M. Mirza, D. Xiao, A. Courville, Y. Bengio, An empirical investigation of catastrophic forgetting in gradient-based neural networks, 2013, arXiv preprint arXiv:1312.6211.

[121] I. Khamassi, M. Sayed-Mouchaweh, M. Hammami, K. Ghédira, Discussion and review on evolving data streams and concept drift adapting, Evol. Syst. 9 (1) (2018) 1–23.

[122] G.I. Webb, R. Hyde, H. Cao, H.L. Nguyen, F. Petitjean, Characterizing concept drift, Data Min. Knowl. Discov. 30 (4) (2016) 964–994.

[123] Y. Zhang, Q. Yang, A survey on multi-task learning, IEEE Trans. Knowl. Data Eng. (2021).

[124] R. Caruana, Multitask learning, Mach. Learn. 28 (1) (1997) 41–75.

[125] M.S. Hammoodi, F. Stahl, A. Badii, Real-time feature selection technique with concept drift detection using adaptive micro-clusters for data stream mining, Knowl.-Based Syst. 161 (2018) 205–239.

[126] A. Dries, U. Rückert, Adaptive concept drift detection, Stat. Anal. Data Min. ASA Data Sci. J. 2 (5–6) (2009) 311–327.

[127] J. Shao, Z. Ahmadi, S. Kramer, Prototype-based learning on concept-drifting data streams, in: Proceedings of the 20th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 2014, pp. 412–421.

[128] J. Gama, P. Medas, G. Castillo, P. Rodrigues, Learning with drift detection, in: Brazilian Symposium on Artificial Intelligence, Springer, 2004, pp. 286–295.

[129] M. Baena-Garcıa, J. del Campo-Ávila, R. Fidalgo, A. Bifet, R. Gavalda, R. Morales-Bueno, Early drift detection method, in: Fourth International Workshop on Knowledge Discovery from Data Streams, vol. 6, 2006, pp. 77–86.

[130] D.M. Manias, I. Shaer, L. Yang, A. Shami, Concept drift detection in federated networked systems, 2021, arXiv preprint arXiv:2109.06088.

[131] A. Bifet, R. Gavalda, Adaptive learning from evolving data streams, in: International Symposium on Intelligent Data Analysis, Springer, 2009, pp. 249–260.

[132] A. Bifet, G. Holmes, B. Pfahringer, R. Gavalda, Improving adaptive bagging methods for evolving data streams, in: Asian Conference on Machine Learning, Springer, 2009, pp. 23–37.

[133] I. Frias-Blanco, J. del Campo-Ávila, G. Ramos-Jimenez, R. Morales-Bueno, A. Ortiz-Diaz, Y. Caballero-Mota, Online and non-parametric drift detection methods based on Hoeffding's bounds, IEEE Trans. Knowl. Data Eng. 27 (3) (2014) 810–823.

[134] K. Nar, O. Ocal, S.S. Sastry, K. Ramchandran, Cross-entropy loss and low-rank features have responsibility for adversarial examples, 2019, arXiv preprint arXiv:1901.08360.

[135] L. Feng, S. Shu, Z. Lin, F. Lv, L. Li, B. An, Can cross entropy loss be robust to label noise? in: IJCAI, 2020, pp. 2206–2212.

[136] Y. Ho, S. Wookey, The real-world-weight cross-entropy loss function: Modeling the costs of mislabeling, IEEE Access 8 (2019) 4806–4813.

[137] Z. Zhang, M.R. Sabuncu, Generalized cross entropy loss for training deep neural networks with noisy labels, in: 32nd Conference on Neural Information Processing Systems, NeurIPS, 2018.

[138] X. Li, L. Yu, D. Chang, Z. Ma, J. Cao, Dual cross-entropy loss for small-sample fine-grained vehicle classification, IEEE Trans. Veh. Technol. 68 (5) (2019) 4204–4212.

[139] D. Rolnick, A. Ahuja, J. Schwarz, T.P. Lillicrap, G. Wayne, Experience replay for continual learning, 2018, arXiv preprint arXiv:1811.11682.

[140] A. Chaudhry, M. Rohrbach, M. Elhoseiny, T. Ajanthan, P.K. Dokania, P.H. Torr, M. Ranzato, Continual learning with tiny episodic memories, 2019.

[141] H. Shin, J.K. Lee, J. Kim, J. Kim, Continual learning with deep generative replay, 2017, arXiv preprint arXiv:1705.08690.

[142] G.M. Van de Ven, A.S. Tolias, Generative replay with feedback connections as a general strategy for continual learning, 2018, arXiv preprint arXiv:1809.10635.

[143] J. Kirkpatrick, R. Pascanu, N. Rabinowitz, J. Veness, G. Desjardins, A.A. Rusu, K. Milan, J. Quan, T. Ramalho, A. Grabska-Barwinska, et al., Overcoming catastrophic forgetting in neural networks, Proc. Natl. Acad. Sci. 114 (13) (2017) 3521–3526.

[144] J. Schwarz, W. Czarnecki, J. Luketina, A. Grabska-Barwinska, Y.W. Teh, R. Pascanu, R. Hadsell, Progress & compress: A scalable framework for continual learning, in: International Conference on Machine Learning, PMLR, 2018, pp. 4528–4537.

[145] H. Ritter, A. Botev, D. Barber, Online structured laplace approximations for overcoming catastrophic forgetting, 2018, arXiv preprint arXiv:1805.07810.

[146] J. Serra, D. Suris, M. Miron, A. Karatzoglou, Overcoming catastrophic forgetting with hard attention to the task, in: International Conference on Machine Learning, PMLR, 2018, pp. 4548–4557.

[147] J. Yoon, E. Yang, J. Lee, S.J. Hwang, Lifelong learning with dynamically expandable networks, 2017, arXiv preprint arXiv:1708.01547.

[148] X. He, J. Sygnowski, A. Galashov, A.A. Rusu, Y.W. Teh, R. Pascanu, Task agnostic continual learning via meta learning, 2019, arXiv preprint arXiv:1906.05201.

[149] A. Mallya, D. Davis, S. Lazebnik, Piggyback: Adapting a single network to multiple tasks by learning to mask weights, in: Proceedings of the European Conference on Computer Vision, ECCV, 2018, pp. 67–82.

[150] A. Mallya, S. Lazebnik, Packnet: Adding multiple tasks to a single network by iterative pruning, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2018, pp. 7765–7773.

[151] M. Masana, T. Tuytelaars, J. van de Weijer, Ternary feature masks: zero-forgetting for task-incremental learning, 2020, arXiv preprint arXiv:2001.08714.

[152] A.A. Rusu, N.C. Rabinowitz, G. Desjardins, H. Soyer, J. Kirkpatrick, K. Kavukcuoglu, R. Pascanu, R. Hadsell, Progressive neural networks, 2016, arXiv preprint arXiv:1606.04671.

[153] M.G. Bellemare, Y. Naddaf, J. Veness, M. Bowling, The arcade learning environment: An evaluation platform for general agents, J. Artificial Intelligence Res. 47 (2013) 253–279.

[154] Z. Li, D. Hoiem, Learning without forgetting, IEEE Trans. Pattern Anal. Mach. Intell. 40 (12) (2017) 2935–2947.

[155] C. Wah, S. Branson, P. Welinder, P. Perona, S. Belongie, The caltech-ucsd birds-200–2011 dataset, 2011.

[156] M.-E. Nilsback, A. Zisserman, Automated flower classification over a large number of classes, in: 2008 Sixth Indian Conference on Computer Vision, Graphics & Image Processing, IEEE, 2008, pp. 722–729.