FedDANN: Non-generative adversarial learning for federated settings

June 2023

Savelli Claudio, Spaccavento Bruno, Udrea Florentin-Cristian
Polytechnic University of Turin
Turin, Italy
s317680@studenti.polito.it, s314908@studenti.polito.it, s319029@studenti.polito.it

Abstract

The main issue arising when working in the Federated Learning (FL) setting is distributions shift among different clients. In the following work, we analyse the FL framework and a few of the solutions proposed in the literature for its intrinsic problems.

The base FedAvg algorithm is proposed and analysed in the initial part of the paper in order to have a baseline on the FEMNIST dataset and compare it to the centralised setting. The hyper-parameters of the federated setting were analysed in detail and conclusions were drawn. The distributions shift problem, in particular, was then analysed in detail under two different lights: first related to the change of class distribution and then related to the change in the domains among clients. The first class of problems is formally known as Statistical Heterogeneity and the second as Domain Generalisation.

Finally, we propose the first, to the best of our knowledge, adaptation of the well-known adversarial learning technique DANN [8] to the Federated scenario as a possible domain generalisation solution.

Code available on GitHub at the following link.

Index terms — Federated Learning, Domain Generalisation, Statistical Heterogeneity, Adversarial Learning

1. Introduction

Federated learning (FL) [3] is a machine learning paradigm in which multiple clients cooperate to learn a model under the orchestration of a central server. This is done in order to preserve the privacy of clients, which is the main reason behind the creation of this paradigm, since in order to train the model in a centralised way, it would need access to possibly sensitive client data, which would jeopardise their right to privacy.

Learning data in a federated setting, although it can open the door to countless applications [4] [5], presents new obstacles to be faced such as:

- **Privacy and security:** the main reason behind the conception of the FedAvg algorithm [3], explained earlier.
- **Statistical:** Devices frequently generate and collect data in a non-indipendent and identically distributed (nIID) manner across the network, e.g., in classification tasks. This data generation paradigm, by violating frequently-used independent and identically distributed (IID) assumptions in distributed optimization, adds complexity in terms of modeling, analysis, and evaluation. [6]
- **Systems:** the number of clients in a federated setting is generally larger than the number of data of interest in each client. Moreover, each client may have limitations in terms of computation, communication and availability, and may differ from the others in terms of hardware, power and network connection. [15]

In our work we will focus mainly on the statistical problems in federated learning:

- in section 4: a detailed explanation of the datasets used was made;
- in section 5: an analysis of the two baselines, the centralised one using EMNIST [2] and the federated one using FEMNIST [15];
- in section 6: some proposed methods to tackle the problem of statistical heterogeneity, in particular smart client selection [16] and server-side momentum [6];
- in section 7 : an analysis of the domain generalisation problem by means of leave-one-domain-out tests, and the use of the FedSR algorithm [14];
- finally, in sections 8 and 9, the description of our proposed method to tackle the domain generalization problem by means of non-generative adversarial learning [8] and its results.

2. Related work

Domain generalisation (DG) aims to achieve out-of-distribution generalization by using only source data for model learning [11]. One approach proposed by Liu et al.

[12], called ***FedDG***, specifically created for image segmentation in the medical field, consists of boundary-oriented episodic learning by exchanging multi-source distributions across the frequency space. This method, though, can be seen as a form of data leakage, which might be considered against FL principles.

Another approach, proposed by Zhang et al. [13], called ***FedADG*** tries to align the distributions among different source domains by matching each distribution to a reference distribution, the latter being derivated from a generative adversarial model. Despite the fact that this, unlike the previous approach, does not violate the principle of data privacy, it may be overly complicated to put into practice, and fails in any case to match state-of-the-art centralised methods.

Inspired by this last approach, Nguyen et al. [14] proposed the ***FedSR*** framework, where the model learns a "simple representation" of the data by using an L2-norm regulariser on the representation and Conditional Mutual Information, between the representation and the data given the label, regulariser to encourage the model to learn essential features, common to all domains.

In the context of Domain Alignment, Ganin et al [8] proposed the use of Domain-Adversarial Neural Networks in which source labelled and target unlabelled data are used in order to find features that can best differentiate classes and be indiscriminate in the context of domain-shifting. Even though the ***DANN*** architecture was initially intended for domain alignment (DA), in which both source domain and target domain inputs are used in the training procedure, the intuition behind was applied also to the domain generalisation problem (DG) [9], where only source domain are used for training. However, this approach has been applied in a non-federated setting and is based on the concept of domain adaptation in which there is a strong assumption that the target data is accessible to the model, which is not always the case in practice. Motivated by this, we propose our approach to the problem of domain generalisation in the FL setting through the use of Domain-Adversarial Neural Networks.

In the context of Statistical heterogeneity, the proposals of Cho et al. [16] and Hsu et al. [6] to solve the problem were taken and reproduced. Specifically, the former [16] tried to solve the problem by no longer randomly selecting clients for each round, the latter [6] by adding momentum to the server, creating a dual optimizing procedure both internally for each client and externally for the server at the moment of aggregation.

3. Problem Statement

3.1. FedAvg algorithm

One of the first algorithms proposed was the "Federated Averaging" algorithm, or FedAvg. The goal of the algorithm is to minimise the following objective function:

$$\min_{w \in \mathbb{R}^d} f(w) \quad \text{where} \quad f(w) \stackrel{\text{def}}{=} \frac{1}{n} \sum_{i=1}^{n} f_i(w) \tag{1}$$

For a machine learning problem, we typically take $f_i(w) = l(x_i, y_i; w)$, that is, the loss of the prediction on example (x_i, y_i) made with model parameters w. We assume there are K clients over which the data is partitioned, with $\mathcal{P}_k$ the set of indexes of data points on client k, with $n_k = |\mathcal{P}_k|$. Thus, we can re-write the objective 1 as:

$$f(w) = \sum_{k=1}^{K} \frac{n_k}{n} F_k(w) \quad \text{where} \quad F_k(w) = \frac{1}{n_k} \sum_{i \in \mathcal{P}_k} f_i(w) \tag{2}$$

This algorithm works by choosing a fraction C of the clients (if $C = 1$ then all K clients are taken) each learning round. Once chosen the active clients the server sends them the current model (a model randomly initialised at the beginning), then each client uses its own data to train the model, $w_{t+1}^k \leftarrow w_t^k - \eta \nabla F_k(w_t^k)$ for an E number of epochs and then sends the server the number of data used to train the model, n_k, and the trained model, w_{t+1}^k; finally the server uses this received data to update the overall model $w_{t+1} \leftarrow \sum_{k=1}^{K} \frac{n_k}{n} w_{t+1}^k$. All this for an a priori decided number of rounds, usually a few hundred.

3.2. Statistical heterogeneity

Statistical heterogeneity refers to the presence of variations in the data distributions across different domains or clients in a federated learning setting. It is a problem because it can adversely affect the performance and generalization capability of the federated learning model.

For example, it can introduce bias and unfairness in the model, favoring domains with more representation or can result in poor generalization as the model struggles to capture common patterns across diverse domains.

As can be seen in fig.1, the problem occurs when there are variations in the data distributions across different domains or clients participating in the federated learning setting. In fact, clients use their local data to train the model and these may also have different types of devices or sensors, resulting in variations in data distribution.

Due to the nature of the federated framework, it is not possible to directly compare these distributions for privacy reasons, which therefore raises several challenges in this framework, trying to find techniques that can mitigate this phenomenon and make the model more robust.

3.3. Domain generalisation

In federated learning each client typically represents a different domain, such as different hospitals, companies, or geographical regions. These domains may have variations

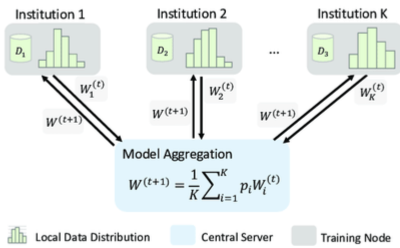

Figure 1. *Example of the Statistical Heterogeneity problem*

in data distributions, feature representations, and data collection processes. The main goal of federated learning is to train a model that can generalize well to unseen data from all the participating domains.

This is where we see the main problem with Domain Generalisation in Federated due to the very nature of the framework we are using. As a matter of fact, it is never possible for any client to see the domains of other clients, and our model will only be able to train on one client and thus domain at a time, which makes federated domain generalisation problem quite unique.

Formally we describe a domain as a set of data $\mathcal{S}^k = \{(\mathbf{x}_i, y_i)\}_{i=1}^n \sim P_{XY}^k$, where $\mathbf{x}_i$ is taken from the input space $\mathcal{X} \subset \mathbb{R}^d$ and y_i is taken from the output space $\mathcal{Y} \subset \mathbb{R}$. Each domain is described by a joint distribution P_{XY}^k. Consequently X and Y denote the corresponding random variables.

In domain generalisation we are given a set of training (source) domains $\mathcal{S}_{train} = \{\mathcal{S}^i | i = 1, ..., M\}$, where the joint distributions between each pair of domain is different: $P_{XY}^i \neq P_{XY}^j \forall i \neq j$. The goal of domain generalization is to learn a robust and generalisable predictive function $h : \mathcal{X} \rightarrow \mathcal{Y}$ from the M training domains to achieve a minimum prediction error on an unseen test domain S_{test} (i.e. S_{test} cannot be accessed during training and $P_{XY}^{test} \neq P_{XY}^i$ for $i \in \{1, ..., M\}$, as described in fig. 2.

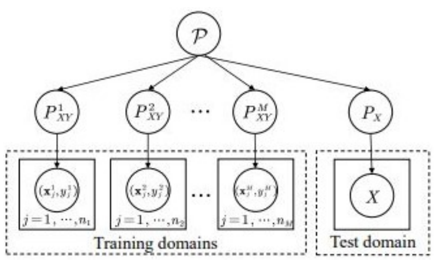

Figure 2. *Example of the Domain Generalisation problem*

3.4. Domain adversarial regularisation

Our proposal seeks to bridge the effects of domain-shifting between clients by using Domain-Adversarial Neural Networks [8], DANN for short, whose architecture is similar to that of a classical CNN with a few differences. The first part of the network, called the feature extractor, consists of the convolutional layers typical of a CNN; two branches branch off from this, the first is the classic label predictor, whose loss function is minimised, the second is the domain classifier, where the aim is to maximise the respective loss function as much as possible. The purpose of these two branches is to train the model to correctly recognise class labels, but also not to be able to distinguish between different domains, in short we want the features found by the feature extractor to be class-accurate, but domain-invariant. Hence the concept of 'Domain-adversarial', as one of the branches seeks to minimise its loss function, while the other seeks to maximise it.

For the sake of simplicity we will consider our neural network to have only one hidden layer, nevertheless the insights we'll get out of the following analysis are useful for more complex networks as well. Therefore we will describe our featuriser as:

$$\mathbf{z} := G_f(\mathbf{x}; \mathbf{W}, \mathbf{b}) = ReLU(\mathbf{W}\mathbf{x} + \mathbf{b}) \tag{3}$$

Similarly we describe the classifier as:

$$\hat{\mathbf{y}} := G_y(\mathbf{z}; \mathbf{V}, \mathbf{c}) = \mathbf{V}\mathbf{z} + \mathbf{c} \tag{4}$$

having $ReLU(x) \stackrel{\text{def}}{=} max(0, x)$. Given a sample $(\mathbf{x_i}, y_i)$ we will estimate the classification loss with the cross entropy loss, described as:

$$\mathcal{L}_y(\hat{\mathbf{y}}_\mathbf{i}, \mathbf{y_i}; \mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) = -\sum_{c=1}^{C_y} \log \frac{\exp(\hat{y}_{i,c})}{\sum_{k=1}^{C_y} \exp(\hat{y}_{i,k})} y_{i,c} \tag{5}$$

having $\hat{y} = G_y(G_f(\mathbf{x_i}; \mathbf{W}, \mathbf{b})); \mathbf{V}, \mathbf{c})$ explains also why $\mathcal{L}_y$ is also a function of $\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}$. Thus the classification's optimisation problem becomes

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y(\hat{\mathbf{y}}_\mathbf{i}, \mathbf{y_i}; \mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c})) \right] \tag{6}$$

By adding the adversarial branch to our neural network we introduce a domain classifier as:

$$\hat{\mathbf{d}} := G_d(\mathbf{z}; \mathbf{U}, \mathbf{t}) = \mathbf{U}\mathbf{z} + \mathbf{t} \tag{7}$$

and the domain classification loss $\mathcal{L}_d(\hat{\mathbf{d}}_\mathbf{i}, \mathbf{d_i}; \mathbf{W}, \mathbf{b}, \mathbf{U}, \mathbf{t})$ also defined as cross entropy loss (thus similarly to (5)), the classification problem (6) becomes a regularised minimization problem:

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y^i(\mathbf{W}, \mathbf{b}, \mathbf{V}, \mathbf{c}) + \lambda \cdot R(\mathbf{W}, \mathbf{b}) \right] \tag{8}$$

Where we define the regularisation term as the maximisation of the domain classification loss, in order to ensure the domain-invariance of the domains in the feature latent space:

$$R(\mathbf{W}, \mathbf{b}) = \max_{\mathbf{U},\mathbf{z}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_d^i(\mathbf{W}, \mathbf{b}, \mathbf{U}, \mathbf{z}) \right] \tag{9}$$

Thus we can express our adversarial optimisation problem in full form as:

$$\min_{\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}} \left[\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_y^i(\mathbf{W},\mathbf{b},\mathbf{V},\mathbf{c}) + \lambda \cdot \max_{\mathbf{U},\mathbf{z}} \left(\frac{1}{n} \sum_{i=1}^{n} \mathcal{L}_d^i(\mathbf{W},\mathbf{b},\mathbf{U},\mathbf{z}) \right) \right] \quad (10)$$

4. Datasets Used

All datasets that were used in the conduct of the experiments for reproducibility purposes are described below. This section seemed important to include as a new dataset (DGFEMNIST) was created for the development of the experiments.

4.1. EMNIST

The EMNIST (Extended MNIST) dataset [2] is a set of handwritten character digits derived from the NIST Special Database 19 and converted to a 28x28 pixel image format and dataset structure that directly matches the MNIST dataset. In particular the EMNIST ByClass is considered, in which 814,255 characters are presents, divided in 62 classes (10 digits, 26 lowercase and 26 uppercase characters).

4.2. FEMNIST

The FEMNIST (Federated Extended MNIST) dataset [15] is built by partitioning the data in Extended MNIST into approximately 3500 clients. The dataset can either be split in a IID (Independent and identically distributed) or nIID (Non Independent and identically distributed) fashion. In the first case, all users have the same underlying distribution of data. In the second case instead, each client is a different writer, so the local datasets differ on both classes and handwriting.

4.3. RotatedFEMNIST

Taking inspiration from [14], the RotatedFEMNIST dataset was created. The only difference to the cited paper is related to the fact that we are in a Federated environment, so instead of transforming 1000 images, 1000 clients were taken into account and one of the 6 transformations proposed in Fig. 3 was applied for all its images (equally distributed, so each transformation is applied to 167 clients out of the approx. 3500 total).

Figure 3. *Example of all the transformations of RotatedFEMNIST on an image taken from FEMNIST*

4.4. DGFEMNIST

Considering that during the experiments the use of RotatedFEMNIST often gave similar results to normal FEMNIST, since the shift domain between one domain and the other is relatively low especially for the first rotations, it was decided to create a new set of transformations for a new dataset that could generate sharper shift domains. The six transformations, observable in FIG. 14, are described below:

1. No transformation.
2. Motion blur.
3. Colour inversion.
4. 45-degree rotation.
5. Random noise.
6. Random translation and rescaling.

The distribution of transformed clients is the same as that followed in RotatedFEMNIST. More examples of transformed images can be seen in appendix A.

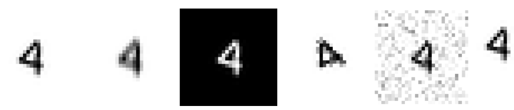

Figure 4. *Example of all the transformations of DGFEMNIST on an image taken from FEMNIST*

5. Experiment setting and benchmarks

The experiments in the following paragraphs, unless explicitly stated otherwise, were carried out with a very simple Convolutional Neural Network (CNN), the architecture of which we report here for the reproducibility of the experiments.

- Two 5x5 convolutional layers with 32 and 64 output channels respectively.
- Each convolutional layer is followed by both a 2x2 max pooling layer and a ReLU activation layer.
- Two fully connected layers, the fist one mapping the input to 2048 output features, and the second one mapping to the number of classes, that is 62 in both EMNIST and FEMNIST.
- Dropout is applied with $p = 0.25$ to all the layers except the last one.

5.1. Centralised Baseline using EMNIST

Before starting to carry out the experiments in the Federated environment using FEMNIST, to make an initial selection of the hyperparameters and to gain familiarity with the dataset, some experiments were first carried out in the centralised environment using EMNIST.

Furthermore, the main aim of these experiments is also to have a centralised baseline, which can be seen as the best result that can be obtained later using the same network in the Federated framework.

The tuning of the hyperparameters was performed on the learning rate and the weight decay, as it was seen that momentum other than the one used ($m = 0.9$) led to a clear decrease in accuracy. The values tested were $lr = \{0.1, 0.01, 0.001\}$ and $wd = \{0, 1\text{e-}4, 1\text{e-}5\}$.

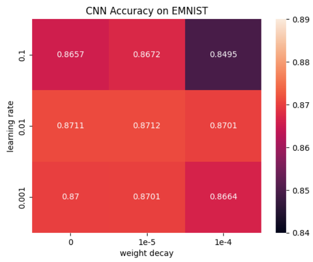

Figure 5. *Results obtained when varying the weight decay and learning rate*

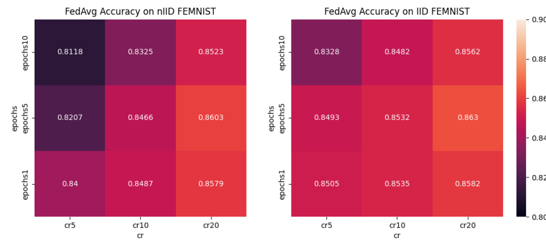

Figure 6. *Results obtained when varying the number of clients per round and epochs per client, for both IID and nIID*

In Fig. 5, one can observe the results obtained, and see that the best hyper-parameters are $lr = 0.01$ and $wd =$ 1e-5, with an accuracy of 0.8712.

5.2. Federated Baseline

For these experiments, and all federated experiments except when explicitly written, Leaf's FEMNIST [15] dataset was used. For Federated's baseline, the learning rate of each individual client was set to $lr = 0.1$, momentum $m = 0.9$, batch size $bs = 64$ and weigth decay $wd =$ 1e-4. The optimiser chosen is SGD. As far as the server is concerned, however, we left $lr = 1$ and $m = 0$ for this initial phase of experimentation.

The hyper-parameters taken into account in this phase of the experiment are the clients selected per round (cr) and the number of training epochs each client performs (ne) in both the IID and nIID cases.

The objective is to find a correct tradeoff between accuracy obtained in the result, the amount of data sent to the server ($\propto cr$) and the computational cost of the devices ($\propto ne$).

As expected, as the number of selected clients per round increases, the accuracy of the model also increases. However, it is important to note that, from the results obtained, the growth of accuracy in relation to the number of users appears to be logarithmic (as it is necessary to double the number of clients in order to have linear growth in accuracy), as can be seen in Fig. 6. On the other hand, by increasing the number of ne, the model does not necessarily improve in accuracy. As a matter of fact, since we do not have a centralised dataset but only a few data points within each individual client, we only consider the latter in each epoch, which has a great influence on the SGD step.

In fact, the individual clients contain much less information than the respective batches in the centralised model, so we would have an effect similar to iterating over the same batch several times before moving on to the next one.

As mentioned before, we therefore tried to find a valid tradeoff between the results obtained and the costs attached, and for this reason for the experiments to follow, $cr = 10$ and $ne = 1$ were set as hyperparameters. With these hyperparameters, selecting 10 clients per round and having each of them do an epoch, the accuracies obtained are 0.8535 for IID and 0.8487 for nIID.

6. Statistical Heterogeneity

In this chapter, a couple of methods proposed in the literature to attempt to solve the problem of Statistical Heterogeneity are proposed.

6.1. Smart client selection

The FedAvg algorithm, as described in 3.1 chooses a fixed number clients each round by randomly picking from the available set. This scheme is known as *unbiased client selection*.

Yae Jee et al. in [16] have shown how *biased client selection* plays an important role in heterogeneous environments (nIID scenario). In particular they explore how biasing the client selection towards those whose local loss is higher archieves better error convergence. In particular the paper introduces the *power of choice* client selection and it is managed in two steps:

- a non-biased client selection by choosing d clients within the entire client pool, creating the client set A
- a biased client selection by choosing $m \leq d$ clients having the highest local loss within the client set A

As elaborated in the cited paper, a larger d value indicates a bigger selection skew, thus d increases error convergence speed at the risk of higher error floor.

For out set of experiments we kept $m = 10$ and changed $d = \{20, 50, 80, 100\}$. As shown in table 1, it is evident how the power of choice client selection archieves greater results when hyperparameter d is appropriately chosen, yet increasing it too much leads to a non negligible higher error floor. In fact, while having $d = 20$ increases the performance by about 1% in both IID and nIID scenarios, further increasing d to higher values not only does not help in boosting performances, but it can yield lower performances than the unbiased client selection.

Table 1. Power of choice results

	$d = 20$	$d = 50$	$d = 80$	$d = 100$	$random$
nIID	**0.8588**	0.8571	0.8536	0.8520	0.8487
IID	**0.8623**	0.8569	0.8566	0.8499	0.8535

6.2. Server-side momentum

Referring back to the experiments conducted by [6], [7] and noticing a slow initial growth in the accuracy of the model, it was decided to try implementing the momentum server, which should accelerate the initial growth of the model. Therefore, instead of simply performing an aggregation at the end of each round, it was decided to take the weights obtained from each client to calculate a server step, performed with $lr = 1$ and momentum $\{0.5, 0.6, 0.7, 0.9\}$. By hyperparameter tuning the selected server momentum (sm) was $sm = 0.6$.

It was seen that the results obtained at the end of the 1000 standard rounds were slightly better than those obtained without using the momentum server (0.859 vs. 0.849 obtained without using sm). Still, difference can be seen in the achievement of convergence speed between the two settings.

In fact, in FIG. 7 the results are shown taking into consideration only 100 rounds and testing the validation every 10. At the end of the 100 rounds, tests were also carried out on the test set considering both settings, obtaining a very considerable difference: while 0.755 was obtained with the normal method, with the momentum server a final accuracy of 0.807 ($+5.2\%$) was obtained.

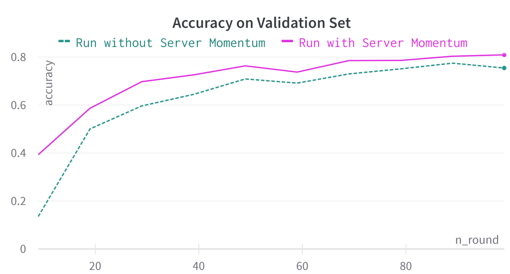

Figure 7. *Comparison accuracy assessed every 10 rounds on the validation set with and without the use of the momentum server*

7. Domain generalisation

In this section, the main problems associated with Domain Generalisation (DG) in a Federated environment will be shown and analysed and FedSR will be seen in depth. To do this, the two datasets RotatedFEMNIST and then DGFEMNIST are taken into consideration. For each of these datasets, tests were carried out either by transforming only 1000 clients out of the almost 3500 or so, following [14], or by transforming all of them.

7.1. Centralised and Federated Baseline

To better understand how federated suffer more from the domain shift problem than a centralised environment, some tests were carried out. For these experiments, the transformations (both RotatedFEMNIST and DGFEMNIST) have been performed on the clients, they are divided into train, validation and test sets and the normal process is carried out. We are still not talking about DG since we train on all domains.

It is emphasised that since most of the clients have no transformation applied in (1000), the results are not too far from the normal FEMNIST. For this reason, the same baseline was also made by transforming all clients. As expected, in this case the accuracies obtained are lower, as there is a greater domain shift between all clients. The results obtained can be seen in the table 2.

Table 2. Centralised and Federated Baseline with RotatedFEMNIST and DGFEMNIST

	$RF(1000)$	$RF(ALL)$	$DGF(1000)$	$DGF(ALL)$
Centralised	0.878	0.846	0.880	0.867
Federated	0.851	0.802	0.863	0.841

7.2. Leave-one-domain-out

With the Leave-one-domain-out strategy, the focus is shifted to DG. This is inspired by and described in more detail in Sec. 4.2 of [14].

In Leave-one-domain-out one of the six transformations applied is disregarded from the dataset in the training, validation and test phase. At the end of the rounds, after carrying out the same procedure as in the baseline on all clients minus those with the transformation excluded, is a second test carried out on the left-out domain, never before seen by the federated model, as per the definition of DG.

The results regarding the accuracy obtained with each left-out domain for both datasets can be seen in section 9. As can easily be imagined, the results obtained vary greatly depending on the left-out domain, having higher accuracy when the domain is 'similar' to those used in the training, and lower ones moving away from the domains used in the training. This can be easily observed in Fig. 8, where, as the rotation increases, the accuracy decreases.

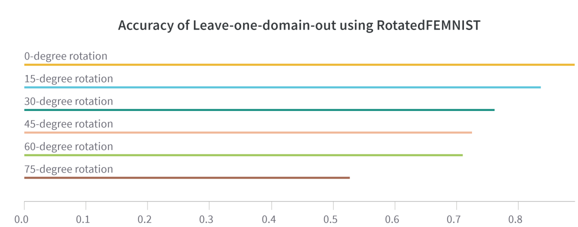

Figure 8. *Accuracies obtained for each left-out domain using RotatedFEMNIST dataset*

7.3. Federated Simple Representation (FedSR)

The FedSR [14] method attempts to approach the problem of domain generalisation in the FL setting. To achieve this, the authors of the above algorithm propose to learn a "simple representation" of the data, through the use of two regularisation methods, specifically the L2 regulariser, named l^{L2R}, and the conditional mutual information, between the data and the representation given the label, regulariser, named l^{CMI}. The final local objective function of each client i is:

$$\overline{f_i} + \alpha^{L2R} l_i^{L2R} + \alpha^{CMI} l_i^{CMI} \tag{11}$$

These two regularisers aim to limit the amount of information the data representation can contain, in order to ignore spurious correlations and lead to a better generalisation of unseen target domains, while at the same time respecting FL principles, as there is no exchange of data between different clients. By doing hyperparameter tuning the selected L2R coefficient is $\alpha^{L2R} = 0.01$ and the selected CMI coefficient is $\alpha^{CMI} =$ 1e-4. The results regarding the accuracy obtained with each left-out domain for both datasets can be seen in the tables in section 9.

8. Proposed method

In order to implement equation (10) the feature extractor are connected through a gradient reversal layer (GRL), which leaves the inputs unchanged during forward propagation and reverses the gradient during backpropagation by multiplying it by a negative scalar. In our particular case we mantained the CNN architecture described in section 5 and used the last layer prior to the classification as feature representation layer (having 2048 features). The domain classifier branch was formed by two fully connected layers: going from 2048 features to 1024 and then to $n\_domains$. Its negative loss, which was backpropagated to the featuriser, was also multiplied by a regularisation factor $\lambda = 0.01$. An example of DANN architecture implementation can be seen in fig. 8

Applying DANN to the federated setting came with a new set of challanges, particularly regarding the distance of the domains and their distribution among clients. This is why a total of 4 different settings were tested, using the two datasets (RotatedFEMNIST and DGFEMNIST), and for each of which we first transformed only 1000 clients following [14], and then all clients for the same reasons as shown in 7.1.

9. Results

In the following section, the main results obtained from the experiments are reported and commented on, comparing FedAvg, FedSR and our FedDANN in the leave-one-domain-out framework.

As can be seen, on average DANN performs better in these Domain Generalisation problems.

In particular, as expected, the results are even better when dealing with larger domain shifts and when more clients are being transformed. In fact, in these environments mentioned above, an adversarial approach to Domain Generalisation and consequently feature extraction independent from the domain is more effective.

Given the results obtained, to ensure that our DANN was working correctly, the Domain Classifier was studied in detail and compared with the Label Classifier.

Plotting the loss of both classifiers, the expected results were obtained: while the classifier loss continued to fall the domain classifier loss initially decreased, then it bounced off and started zigzagging around a certain stable value. This happened as it initially understood how to classify the domains, so the featuriser had to adapt its representations in order to make them more domain-invariant.

Furthermore, an even more interesting result can be seen in Fig. 11. In fact, during the training phase, domains were classified by our network, and it is possible to see how the results obtained are random guesses. There is an average accuracy of $1/5 = 0.2$ when there are 5 different domains

Table 3. RotatedFEMNIST (1000) - L1O accuracies

	0°	15°	30°	45°	60°	75°	Avg.
FedAvg	**0.891**	0.836	**0.761**	**0.725**	0.710	0.527	0.741
FedDANN	0.888	**0.842**	0.756	0.723	**0.711**	**0.543**	**0.744**
FedSR	0.876	0.811	0.724	0.696	0.664	0.473	0.707
Centralised	*0.886*	*0.848*	*0.778*	*0.730*	*0.716*	*0.551*	*0.752*

Table 4. RotatedFEMNIST (ALL) - L1O accuracies

	0°	15°	30°	45°	60°	75°	Avg.
FedAvg	0.681	**0.784**	**0.779**	0.774	**0.795**	**0.605**	0.736
FedDANN	**0.685**	**0.784**	**0.779**	**0.776**	**0.795**	**0.605**	**0.737**
FedSR	0.658	0.753	0.757	0.752	0.763	0.579	0.710
Centralised	*0.711*	*0.795*	*0.796*	*0.825*	*0.827*	*0.704*	*0.786*

Table 5. DG FEMNIST (1000) - L1O accuracies

	$D1$	$D2$	$D3$	$D4$	$D5$	$D6$	Avg.
FedAvg	**0.895**	0.775	0.079	0.245	**0.826**	0.146	0.494
FedDANN	**0.895**	**0.801**	**0.100**	0.233	**0.826**	0.133	**0.498**
FedSR	0.883	0.702	0.037	**0.264**	0.813	**0.153**	0.475
Centralised	*0.881*	*0.735*	*0.073*	*0.330*	*0.855*	*0.225*	*0.517*

Table 6. DG FEMNIST (ALL) - L1O accuracies

	$D1$	$D2$	$D3$	$D4$	$D5$	$D6$	Avg.
FedAvg	0.841	0.701	0.015	**0.232**	0.820	0.126	0.456
FedDANN	**0.842**	**0.708**	**0.051**	**0.232**	**0.822**	0.131	**0.464**
FedSR	0.825	0.632	0.043	0.221	0.787	**0.134**	0.440
Centralised	*0.864*	*0.707*	*0.053*	*0.267*	*0.753*	*0.179*	*0.471*

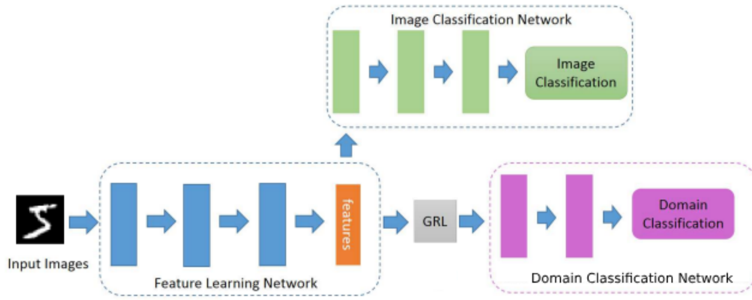

Figure 9. *An example of DANN architecture implementation*

in the training phase, and $1/6 = 0.16$ when there are 6 domains. It is emphasised that with left-out 0 the domains being trained are still 6 when transforming 1000 clients since all images belonging to the non-transformed clients are naturally the same as those in which a rotation of 0 degrees is applied.

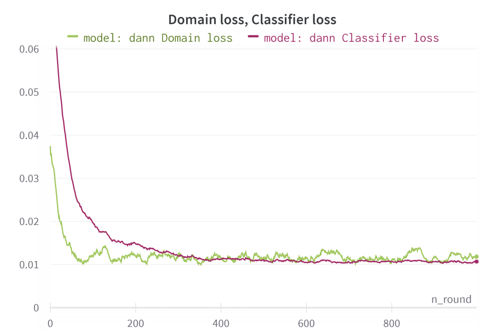

Figure 10. *Smoothed label classifier loss and domain classifier loss (smoothing factor 0.9) taken from DGFEMNIST (1000)*

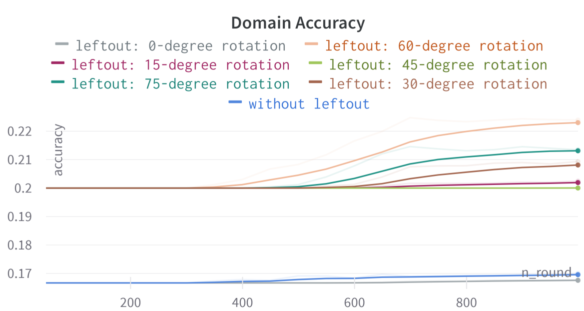

Figure 11. Smoothed domain accuracy get in RotatedFEMNIST with 1000 clients transformed

10. Conclusions

The general analysis of the Federated Learning framework, especially related to Domain Generalisation issues, has been analysed quite comprehensively within the paper. Furthermore, the use for the first time, as far as we know, of a non-generative adversarial network within this framework not only proved to be interesting but also led to satisfactory results exceeding both the results obtained with the normal FedAvg and FedSR.

Obviously, this improvement is achieved in exchange for an additional piece of information, related to the fact that it is necessary to know in advance the domain of the image one wishes to classify. This additional information, on the other hand, is easily obtainable in certain circumstances, for example, when a photo is taken, some information is saved with it that can be used in the categorisation of the domain, such as the time and place where the photo was taken.

In addition, there may be some particular environments, not considered within our paper, where it is possible to use our DANN architecture in some other way. In fact as already anticipated this architecture is designed for Domain Alignment problems, which can still occur in the Federated environment. An example may be in the medical domain: Hospitals serve as establishments harboring an abundance of patient data, critical for predictive healthcare endeavors (Domain Adaptation challenge). Nevertheless, hospitals operate within the confines of rigorous privacy protocols, which elucidates the widespread adoption of Federated Learning [4].

In addition, more complex architectures could have been tried for conducting the experiments and more tests could have been performed but we were limited by the available computing power. In fact, all tests were carried out with a very simple CNN, and even in the case of the DANN architecture we merely added a bifurcation to our starting CNN, adding the domain classifier, without performing a careful analysis on the network itself.

To conclude, the results obtained overall are still very satisfactory, although there is certainly still room for improvement. In fact, we hope that this could be an interesting starting point for evaluating this methodology applied to the federated learning framework.

References

[1] Google AI Blog, Federated Learning: Collaborative Machine Learning without Centralized Training Data.

[2] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. ”EMNIST: an extension of MNIST to handwritten letters”. arXiv preprint arXiv:1702.05373, 2017.

[3] McMahan, Brendan et al. “Communication-Efficient Learning of Deep Networks from Decentralized Data.” Proceedings of the 20th International Conference on Artificial Intelligence and Statistics, PMLR 54: 1273-1282, (2017).

[4] Li, Tian, et al. “Federated Learning: Challenges, Methods, and Future Directions.” IEEE Signal Processing Magazine 37.3 (2020): 50-60.

[5] Kairouz, Peter, et al. “Advances and Open Problems in Federated Learning.” arXiv preprint arXiv: 1912.04977 (2019).

[6] Tzu-Ming Harry Hsu, Hang Qi, and Matthew Brown. “Measuring the effects of non-identical data distribution for federated visual classification.”, 2019.

[7] Hsu TM.H. et al. “Federated Visual Classification with Real-World Data Distribution. European Conference on Computer Vision . ECCV 2020. Lecture Notes in Computer Science, vol 12355. Springer, Cham.

[8] Yaroslav Ganin, Evgeniya Ustinova, Hana Ajakan, Pascal Germain, Hugo Larochelle, François Laviolette, Mario Marchand, Victor Lempitsky, ”Domain-Adversarial Training of Neural Networks”. arXiv:1505.07818 [stat.ML].

[9] Isabela Albuquerque, et al.: ”Adversarial target-invariant representation learning for domain generalization”. arXiv preprint arXiv:1911.00804 (2020)

[10] Caldarola, D., Caputo, B., & Ciccone, M. (2022, October). “Improving generalization in federated learning by seeking flat minima.” In Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXIII (pp. 654-672). Cham: Springer Nature Switzerland.

[11] Zhou, K., Liu, Z., Qiao, Y., Xiang, T., & Loy, C. C. (2021). “Domain generalization in vision: A survey.” IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2022.

[12] Liu, Quande, et al. ”FedDG: Federated domain generalization on medical image segmentation via episodic learning in continuous frequency space.” Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2021.

[13] Zhang, Liling, et al. ”Federated learning with domain generalization.” arXiv preprint arXiv:2111.10487 (2021).

[14] Nguyen, A. Tuan, Philip Torr, and Ser-Nam Lim. ”FedSR: A Simple and Effective Domain Generalization Method for Federated Learning.” Advances in Neural Information Processing Systems. 2022.

[15] Caldas, Sebastian, et al. ”Leaf: A benchmark for federated settings.” Workshop on Federated Learning for Data Privacy and Confidentiality (2019).

[16] Cho, Yae Jee, Jianyu Wang, and Gauri Joshi. ”Client Selection in Federated Learning: Convergence Analysis and Power-of-Choice Selection Strategies.”, arXiv preprint arXiv:2010.01243 (2020).

Appendix A - More examples of DGFEMNIST

Other transformed images have been included in this appendix to allow the reader a better understanding of them, especially of the transformation linked to random translation and rescaling.

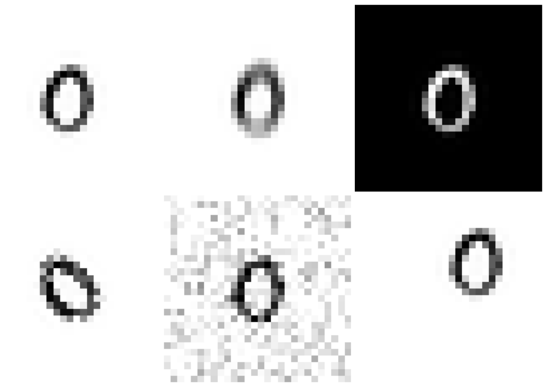

Figure 12. *Example of all the transformations of DGFEMNIST on the number '0'*

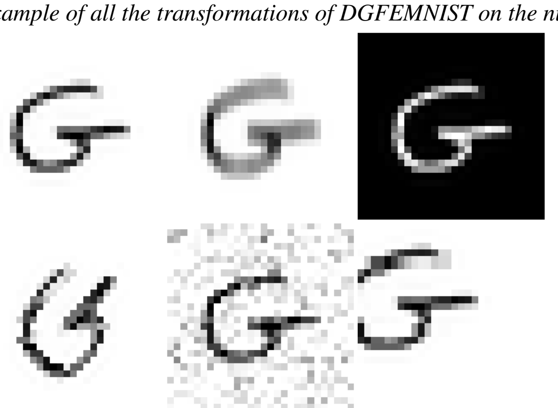

Figure 13. *Example of all the transformations of DGFEMNIST on the letter 'G'*

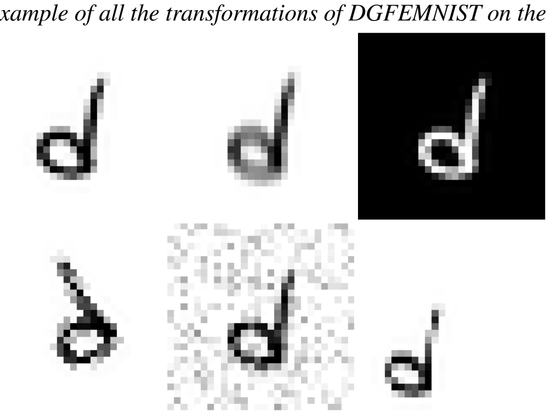

Figure 14. *Example of all the transformations of DGFEMNIST on the letter 'd'*

Appendix B - Overall Accuracies obtained on test sets of Leave-one-domain-out experiments

In the following appendix, all results obtained on the test set at the end of 1000 rounds during the leave-one-domain-out experiments have been reported for completeness.

As can be seen, the results obtained are on average much higher and similar to those obtained with the baseline without any left-out since we are not in a domain generalisation problem.

Table 7. RotatedFEMNIST (1000) - Test set Accuracies

	0°	15°	30°	45°	60°	75°	Avg.
FedAvg	0.848	0.847	0.848	0.850	0.854	0.855	0.850
FedDANN	0.842	0.846	0.846	0.849	0.854	0.854	0.849
FedSR	0.833	0.866	0.829	0.835	0.839	0.838	0.840
Centralised	*0.878*	*0.880*	*0.880*	*0.880*	*0.882*	*0.882*	*0.880*

Table 8. RotatedFEMNIST (ALL) - Test set Accuracies

	0°	15°	30°	45°	60°	75°	Avg.
FedAvg	0.671	0.790	0.803	0.807	0.806	0.807	0.781
FedDANN	0.674	0.790	0.805	0.806	0.806	0.807	0.781
FedSR	0.646	0.756	0.774	0.778	0.780	0.778	0.752
Centralised	*0.718*	*0.830*	*0.849*	*0.857*	*0.857*	*0.855*	*0.828*

Table 9. DG FEMNIST (1000) - Test set Accuracies

	$D1$	$D2$	$D3$	$D4$	$D5$	$D6$	Avg.
FedAvg	0.856	0.860	0.855	0.858	0.861	0.862	0.859
FedDANN	0.857	0.859	0.857	0.859	0.859	0.859	0.858
FedSR	0.844	0.843	0.843	0.849	0.849	0.850	0.846
Centralised	*0.879*	*0.882*	*0.880*	*0.878*	*0.882*	*0.887*	*0.881*

Table 10. DG FEMNIST (ALL) - Test set Accuracies

	$D1$	$D2$	$D3$	$D4$	$D5$	$D6$	Avg.
FedAvg	0.827	0.837	0.839	0.847	0.828	0.841	0.837
FedDANN	0.828	0.838	0.841	0.848	0.830	0.842	0.838
FedSR	0.810	0.818	0.818	0.830	0.807	0.823	0.818
Centralised	*0.855*	*0.868*	*0.871*	*0.871*	*0.868*	*0.833*	*0.861*