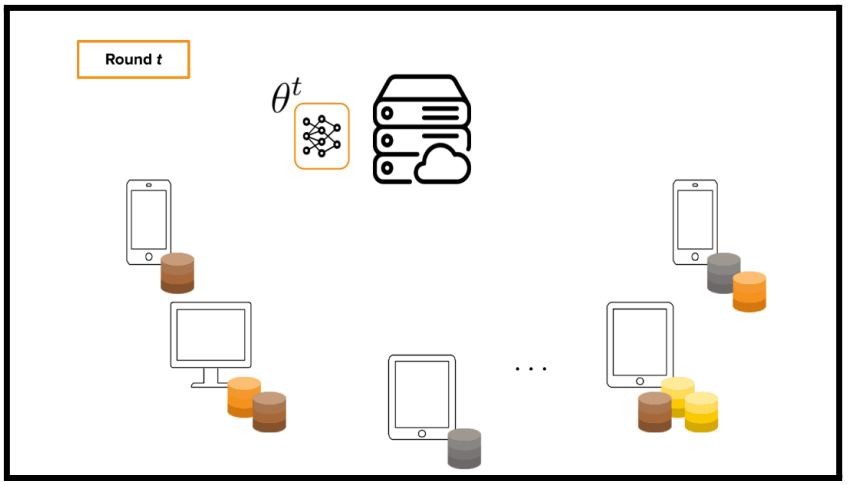
# Towards Real World Federated Learning

## Project Summary

**TAs:** Debora Caldarola, Eros Fanì




Many data nowadays cannot be used by traditional Machine Learning (ML) approaches because of their sensitive and privacy-protected nature, even if their use could significantly improve our models' performance. A concrete example is provided by medical data: doctors may use the knowledge from a trained AI model, yet patient data cannot be collected because it is inherently private. Introduced by Google, **Federated Learning** is an ML scenario aiming to learn from privacy-protected data without violating the regulations in force. Within a client-server architecture, a server-side global model has to be learned by leveraging the clients' data, without breaking their privacy. This is achieved by never giving the server direct access to the data and transferring the trained model parameters.

Let us now imagine the following use case. Alice likes traveling and taking many pictures of the places she visits with her smartphone. On the other hand, John is a professional photographer, so he rarely uses his phone and mainly relies on his camera. Martha is an influencer showing Rome to her followers, so her phone is packed with pictures of that city. All of them would like to have access to an AI model able to recognize the place in which their photos were taken. Due to the private nature of their pictures, they need a federated scenario. Locally, the model will have access to Alice's pictures of many places, almost no photos on John's phone, and Martha's pics from Rome. ***What happens when a model is trained on so differently distributed data?***

In addition, the pictures could be taken in the daytime or at night, during sunny or rainy days

*i.e.* with different weather and light conditions, and so on. All these variables make learning harder. That implies additional issues arise when moving towards more realistic scenarios, which have to be explicitly addressed. In this case, ***can the model generalize to different and potentially new visual shifts, i.e. domains?***

## Projects overview

The goal of this project is to become familiar with the Federated Learning (FL) framework and the issues arising when facing more realistic scenarios.

### Track 1

#### TA: Debora Caldarola

- Task: **image classification**. The goal is to learn to distinguish numbers and letters, written by several users having different handwriting.
- Challenges: heterogeneously distributed data and generalization to unseen domains (*e.g.*, night vs day)

Main goals:

- To become familiar with the standard federated scenario and its state of the art
- To understand the real-world challenges related to statistical heterogeneity and domain shift across clients
- To replicate the experiments detailed in the following sections
- To implement and test your contribution for solving one of the highlighted issues

**You will be provided with both existing code and federated datasets to start running the first experiments. *Details follow.***

# TRACK 1: Federated image classification & Domain generalization

## STEPS

### Become familiar with the literature and the state-of-the-art

Before starting the project, you should take some time to study and get familiar with the federated scenario, its algorithms, challenges and the main proposed solutions. This will also help you understand what to expect from the experiments you will be asked to do. Those are the main references you must study:

1. [1][2][3] to understand the standard FL setting

2. [4][5] to get a more comprehensive view of the issues arising in FL and its current state of the art
3. [6][7][8] for the effects of data statistical heterogeneity
4. [9] for understanding domain generalization (DG) and the main techniques introduced to address it
5. [10][11][12][13] study DG in FL

If you wish to delve into this topic, you will find a more detailed bibliography at the end of this document, containing the references to the latest trends in the FL literature.

## Codebase

### Repository

The starting codebase for your experiments can be found at https://github.com/Erosinho13/MLDL23-FL-project**.** You can download it as:

```
git clone https://github.com/Erosinho13/MLDL23-FL-project
```

The code is organized as follows:

- in `data/` you will download the dataset. Instructions can be found in the following section
- the `datasets/` folder contains utilities for handling the dataset
- in `utils/` you can find some useful functions
- the `main.py` file orchestrates the federated training
- in `client.py` and `server.py` you can find the classes containing useful functions for local training and server-side orchestration and models aggregation respectively.

**Please, be sure to be familiar with the code and have a good understanding of its main parts *before* starting your experiments.**

### Environment setup

Install all needed packages using Anaconda by running

```
conda env create -f mldl23fl.yml
```

### Useful references

All the experiments will be run using **PyTorch**. The official documentation together with useful tutorials can be found here.

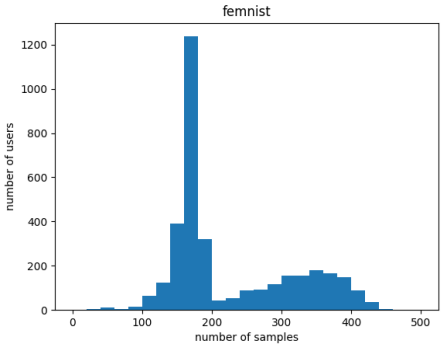
In addition, we suggest you use **Weights and Biases** for keeping track of your experiments and plotting the results. You first need to set up your team account and then install the `wandb` package and import it in the project. Detailed setup instructions can be found here.

## Dataset

This project is based on the **FEMNIST** (Federated Extended MNIST) [14] dataset, which is built by partitioning the data in Extended MNIST [16, 15] based on the writer of the digit/character. FEMNIST contains 28x28 images distinguished over **62 classes**: digits from 0 to 9; 26 lowercase and 26 uppercase characters of the English alphabet.

Some additional **statistics** follow:

- 3,500 users
- 80,5263 samples (total)
- 226.83 samples per user (mean)
- num_samples per user (std): 88.94
- num_samples per user (std/mean): 0.39




The dataset can either be split in a **non-i.i.d.** or **i.i.d.** fashion:

- In the i.i.d. sampling scenario, each datapoint is equally likely to be sampled. Thus, all users have the same underlying distribution of data.
- In the non-i.i.d. sampling scenario, the underlying distribution of data for each user is consistent with the raw data. Since we assume that data distributions vary between user in the raw data, we refer to this sampling process as non-i.i.d. In this scenario, each client is a different writer, so **the local datasets differ on both classes and handwriting**. [14]

To download the dataset, go inside `data/femnist/` and **execute the processing file** as

```
./preprocess.sh -s niid --sf 1.0 -k 0 -t sample
```

and

```
./preprocess.sh -s iid --sf 1.0 -k 0 -t sample
```

to obtain both the iid and non-iid distributions of the dataset among clients.

- `-s`: sampling (i.i.d. or non-i.i.d.)
- `-sf`: dataset fraction (1.0 means full sized dataset)
- `-k`: minimum samples per client
- `-t`: 'sample' to partition each user's samples into train-test groups

The script outputs .json files, each containing 3 keys:

1. `'users'`: a list of users ID
2. `'num_samples'`: a list of the number of samples for each user
3. `'user_data'`: a dictionary having the user ID as key and their respective data as value. Data is a list of images, with each image represented as a size-784 integer list (flattened from 28 by 28)

You can find the created files under

- IID: `data/femnist/data/iid/train` and `data/femnist/data/iid/test`
- non-IID: `data/femnist/data/niid/train` and `data/femnist/data/niid/test`

The function `read_femnist_data()` in `main.py` allows you to read the data from the created .json files.

Run the script in `./stats.sh` to obtain statistics on the dataset.
Write the script to obtain some additional statistics on the distribution of classes among users:

- How many classes do users see on average?
- And with which standard deviation?

## Neural Network Architecture

Taking into account both the complexity of the task and the available GPU resources on Colab, you are asked to run your experiments on a simple Convolutional Neural Network (CNN) of the form:

1. Two 5x5 convolutional layers with 32 and 64 output channels respectively
2. Each convolutional layer is followed by both a 2x2 max pooling layer and an activation layer (*e.g.* ReLU)
3. Two fully connected layers, the fist one mapping the input to 2048 output features, and the second one mapping to the number of classes

The network is the same described in [14].

Create the file containing the network architecture definition (*e.g.* `cnn.py`) in `models/` and import the class in `main.py`. According to the name you choose for your CNN, add the model initialization to the function `model_init()` in `main.py`, as done for ResNet18.

## Centralized training: define the upper bound

The performance of any model trained in a federated fashion is upper bounded by the results obtained in the centralized setting, *i.e.* the standard one. Therefore, as a first step, you need to define that upper bound by training the model in a centralized way **until convergence**. The specifics are as follows:

- **Model**: CNN described above
- **Dataset**: EMNIST (Extended MNIST) [14]
- **Task**: classification on 62 classes
- **Optimizer**: Stochastic Gradient Descent
- **Metric**: Accuracy on test set

You can use the hyperparameters described in the Appendix in [14] as a starting point for finding the best training hyperparameters (*e.g.* learning rate, batch size, momentum, weight decay, epochs, etc).

You should obtain a **test accuracy** of around **89%**.
***How did you finetune and choose the hyperparameters?***
***Did you try different seeds? Why is the testing of different seeds a best practice?***

## Test the federated baseline

It is now time to switch to the federated framework!

### Run your first experiments

1. The first step is to **add your implementation of the CNN** under `models/`. Update the function `model_init()` in `main.py` accordingly. All the experiments will be run using this architecture.
2. Implement the key functions for running the standard FedAvg experiment (all functions to be implemented are identified with a **#TODO**):
    a. Server-side: fill in the function selecting the random fraction of clients to be involved for training for that round and the FedAvg function for aggregating the clients' updates in `server.py`.
    b. Client-side: write the function to perform training and testing on local data in `client.py`.
    c. Dataset: define the get_item() function in `datasets/femnist.py`.
3. **Run the FedAvg experiment** with both IID and non-IID versions of FEMNIST. Training hyperparams:
    a. Ideally, the number of rounds should allow the model to reach convergence. You should be able to run **1000** training rounds on Colab.
    b. Local epochs: test {1, 5, 10}. ***What happens when the local number of epochs increases in the IID scenario? And in the non-IID one?***
    c. Local training hyperparams: same as centralized baseline
    d. Server-side optimizer: SGD with learning rate 1 (FedAvg)
    e. Number of clients per round: test {5, 10, 20}. ***What happens when the more clients are involved at each training round in the IID scenario? And in the non-IID one?***

***Which are the most evident issues arising when the data distribution is heterogeneous?***
***How do the results compare with the centralized training?***

### Moving towards realistic scenarios: smart client selection

In realistic scenarios, clients' availability for training may depend on a number of various reasons, such as the network connection, or the time zone, or the moment in which the device is on charge, or battery life etc. This implies that the server has no guarantees the clients will *uniformly* be able to perform local training when chosen.

***What happens when the clients cannot be selected with uniform probability?***
Modify the function selecting the clients involved at each training round, *i.e.* assign a different probability at each client of being selected.

1. Run the FedAvg experiment (both in IID and non-IID scenarios) with 10 clients per round
    - with 10% of clients being selected with probability 0.5 at each round
    - with 30% of clients being selected with probability 0.0001 at each round

Recent works [17] showed how selecting clients in a smarter way, i.e. according to some criterion, can lead to better results in terms of final performance and speed of convergence. For instance, [17] shows that biasing client selection towards clients with higher local loss achieves faster error convergence.

2. Study both the analysis and the method proposed in [17]
3. Implement the FedAvg experiment with Power-of-choice as clients selection strategy
4. Run the experiment with Power-of-choice in the **IID** and **non-IID** scenarios with 10 clients per round varying the value of *d*, which measures the size of the candidate client set and controls the trade-off between convergence speed and solution bias. Test values of *d* in {2, 5, 8, 10}.

***How does the smart client selection behave wrt the random strategy? How does the value of* d *impact the results?***

## Domain Generalization in Federated Learning

So far, we have focused on the issues deriving from the so called *label skew* and *feature shift* across clients, meaning that clients have access to different classes/labels, and the written digits or characters differ in the users' handwritings, resulting in different features (*i.e.* different handwritings) mapped to the same output (*e.g.* all "1"s are labeled as belonging to the class "1" even if written by different users).

In realistic scenarios, the ***domain shift*** plays an important role as well. For instance, we can think of tourists taking pictures of the Colosseum during the day, at sunset or at night: the object depicted is the same, but the performance of our neural network will be heavily affected by the change in the light conditions. Many other examples can be made when thinking of weather conditions (*e.g.* sunny days vs rainy days), viewpoints, instruments (*e.g.* smartphone camera vs professional camera), and so on.

In this section, we will focus on Domain Generalization in FL with the goal of answering the question "*What happens when new domains, e.g.* ***new users****, appear? Is our model able to generalize to* ***unseen target data*** *as well?*".

With this in mind, you will first build a new version of FEMNIST to introduce new domains through data augmentations techniques, then you will analyze the performance of the standard model learnt with FedAvg when facing new domains, and lastly you will implement a state-of-the-art algorithm for addressing DG in FL.

1. Study [12] carefully and understand both their method and the experiment scenario
2. **Build RotatedFEMNIST** as described in Sec. 4.1.: images are rotated counterclockwise with an angle of 0°, 15°, 30°, 45°, 60° and 75° to form six domains, identified as $M_0$,$M_{15}$ ,$M_{30}$ ,$M_{45}$ ,$M_{60}$ ,$M_{75}$ . Select 1000 random clients and divide them in 6 groups. Apply to each group one of the 6 rotations.

   **NB** The authors use MNIST in their experiments, which contains only images of digits going from 0 to 9, while we keep referring to the extended MNIST (classification on 62

classes). The used neural networks differ as well (we keep working on the same network used up to now).

3. **Run the centralized experiment with the new dataset**, without accounting for generalization. We use this result as reference with the goal of understanding how the presence of different domains impacts the performances.
4. **Run the FedAvg experiment with the new dataset**, without accounting for generalization, *i.e.* keep treating the "new" clients as standard clients. We use this result as reference with the goal of understanding how the presence of different domains impacts the performances.
5. Now, it is time to understand how the model built with FedAvg performs in terms of generalization. As described in Sec. 4.2, in the experiments, follow the "leave-one-domain-out" strategy: choose one domain as the target domain, train the model on all remaining domains, and evaluate it on the chosen domain. **Run both the centralized and the FedAvg experiments 6 times, each time changing the target domain**: *what can you see from the results? Is one domain more difficult to learn than others?*
6. **Implement FedSR**.
7. **Run the 6 experiments using FedSR**, each time changing the target domain. *Are results consistent with what you saw using FedAvg?*

## Time for your personal contribution!

Last step: time to address one of the challenges seen so far! You can **either choose to implement one of the solutions from the current literature listed below, or propose your own (*extremely encouraged to pick the last option!*)**.
**Please, before starting coding the experiments for this last step, communicate the chosen state-of-the-art algorithm to, or discuss your idea with your TA!**

Legend: each * denotes the expected level of difficulty of the extension, which will be taken into account when evaluating your project.

Some examples from recent literature:

### Statistical Heterogeneity

- [**] FedDyn [19]
- [**] SCAFFOLD [18]
- [*] FedVC [7]
- [*] FedIR [7]
- [*] Classifier Calibration [20]
- [***] FedSpeed [21]

### Domain generalization

- [***] FedDG [10]
- [**] FedADG [11]
- [***] As shown in [22], domains can also be addressed through clustering: what happens if we assign to each domain/cluster a domain-specific model? And if we ensemble the

predictions of each domain-specific model when testing them? Which clustering algorithms could we use?

- [*] Different domains can be built using other kinds of corruptions, e.g. by increasing the light in the image, or the shadows, or by coloring the images (*e.g.* take a look here). How do more pronounced domain shifts affect the performances?

[*****] **Your own ideas!**

## DELIVERABLES

Once completed all the steps, you need to submit:

1. PyTorch scripts with code for all the steps.
2. A complete PDF report (paper-style). The report should contain the following sections: a brief introduction, description of the main related works, a methodological section describing the used algorithms and their purposes, an experimental section with all the results and discussions, and a final brief conclusion. Follow this link to open and create the template for the report.

## EXAMPLES OF QUESTIONS YOU SHOULD BE ABLE TO ANSWER AT THE END OF THE PROJECT

- How is the standard federated framework setup?
- What was FL born for?
- Which are the main issues arising when the clients' data distribution is unbalanced and heterogeneous?
- Why is the heterogeneous distribution of clients' data an issue? How do local models differ?
- Can you mention a few solutions addressing the problem of statistical heterogeneity? Which are their pros and cons?
- What is Domain Generalization (DG)? And which are the main existing techniques for addressing it?
- How is DG addressed in the FL literature?
- What is the domain shift and why is it a problem in FL? How can it occur?
- Can you think of a few examples of real-world applications of FL?

## REFERENCES

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

[2] 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).

[3] Reddi, Sashank, et al. "Adaptive federated optimization." ICLR (2021).

[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] 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.
[9] 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.
[10] 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.
[11] Zhang, Liling, et al. "Federated learning with domain generalization." arXiv preprint arXiv:2111.10487 (2021).
[12] 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.
[13] Fantauzzo, Lidia, et al. "FedDrive: generalizing federated learning to semantic segmentation in autonomous driving." 2022 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2022.
[14] Caldas, Sebastian, et al. "Leaf: A benchmark for federated settings." Workshop on Federated Learning for Data Privacy and Confidentiality (2019).
[15] Gregory Cohen, Saeed Afshar, Jonathan Tapson, and André van Schaik. EMNIST: an extension of MNIST to handwritten letters. arXiv preprint arXiv:1702.05373, 2017.
[16] Yann LeCun. The MNIST database of handwritten digits. 1998.
[17] 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).
[18] Sai Praneeth Karimireddy, et al. “Scaffold: Stochastic controlled averaging for federated learning.” In *International Conference on Machine Learning*, pages 5132–5143. PMLR, 2020.
[19] Acar, D. A. E., Zhao, Y., Navarro, R. M., Mattina, M., Whatmough, P. N., & Saligrama, V. (2021). “Federated learning based on dynamic regularization.” ICLR Oral (2021).
[20] Luo, Mi, et al. "No fear of heterogeneity: Classifier calibration for federated learning with non-iid data." *Advances in Neural Information Processing Systems* 34 (2021).
[21] Sun, Yan, et al. "Fedspeed: Larger local interval, less communication round, and higher generalization accuracy." ICLR (2023).
[22] Shenaj, Donald, et al. "Learning across domains and devices: Style-driven source-free domain adaptation in clustered federated learning." Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision. 2023.