

HEREDITARY

HetERogeneous sEmantic Data integration for the guT-bRain interplaY

Deliverable 2.11

Federated infrastructure design

This project has received funding from the European Union's Horizon Europe research and innovation programme under grant agreement No GA 101137074. Views and opinions expressed are however those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.



**Funded by
the European Union**

EXECUTIVE SUMMARY

Deliverable 2.11 on “Federated Infrastructure Design” presents the results from M1 to M18 of Task 2.6, which will continue until M42. In this deliverable, we report the design of the Federated Learning (FL) infrastructure and the tests we conducted. The current federated infrastructure is:

- **secure** thanks to the encrypted communication via gRPC/TLS and secure aggregation (SecAgg/SecAgg+);
- **flexible**, because it supports horizontal (same data types) and vertical (different data modalities) FL workflows, including embedding-based vertical FL; and,
- **deployable**, because we containerised via Docker and **reproducible** across HPC (Slurm), cloud (Azure), and manual setups.

These features were validated and refined through two project workshops:

- Workshop 3 (Jan 30 & 31 2025), which focused on horizontal FL with XGBoost and CNNs across local, cloud, and multi-site environments.
- Workshop 4 (May 19 2025), which focused on vertical FL across three distributed data sources (i.e., SURF, UNIPD, and UNITO) using split ALS data modalities.

We have identified a minimal participant configuration that allows access to a server provisioned with 16 vCPUs and 32 GB of RAM; clients can optionally leverage GPUs.

In terms of scaling and monitoring, we are using *Flower Next* with persistent servers, CLI, and Messaging API; basic logging is implemented, and we plan to integrate MLflow/W&B. From a privacy and security perspective our approach combines TLS and SecAgg and we are planning upgrades to Differential Privacy (DP), Trusted Execution Environments (TEEs), and defenses against poisoning.

Additionally, we tested our data approach on the BrainLat dataset (MRI + EEG), though limited by low matched-patient count and data errors, the embedding architecture remains adaptable.

This deliverable verifies **Milestone 6**, “*Data management infrastructure*”, by testing the Federated Data Management Infrastructure on use-cases 1 and 2, including data and involved partners.

DOCUMENT INFORMATION

Deliverable ID	D2.11
Deliverable Title	Federated infrastructure design
Work Package	WP2
Lead Partner	SURF
Due date	30.06.2025
Date of submission	24.06.2025
Type of deliverable	R
Dissemination level	PU

AUTHORS

Name	Organisation
Douwe van der Wal (Author)	SURF
Damian Podareanu (Author)	SURF
Juan Manuel Rodriguez (Contributor)	AAU
Gianmaria Silvello (Internal reviewer)	UNIPD
Anna Romanovych (Contributor)	UNIPD

REVISION HISTORY

Version	Date	Author	Document history/approvals
0.1	01/06/2025	Douwe van der Wal	Introduction, Workshops, Flower updates, Data management
0.2	15/06/2025	Damian Podareanu	Executive summary, formatting and overall edits
0.3	16/06/2025	Gianmaria Silvello	Review, adjustments, and clarifications.
0.4	19/06/2025	Juan Manuel Rodriguez	Added information about privacy, iid data on federated learning
0.5	20/06/2025	Damian Podareanu & Douwe van der Wal	Adding Annexes and references and processed review comments
0.6	22/06/2025	Gianmaria Silvello	Revision and suggestions
0.7	24/06/2025	Damian Podareanu & Douwe van der Wal	Implemented suggestions
1.0	24/06/2025	Anna Romanovych	Finalizing the layout of the deliverable

Views and opinions expressed are, however, those of the author(s) only and do not necessarily reflect those of the European Union. Neither the European Union nor the granting authority can be held responsible for them.

Contents

List of Abbreviations	7
1 Introduction.....	9
1.1 Objectives of Task 2.6	9
2 Workshops on horizontal and vertical FL.....	10
2.1 Explaining horizontal and vertical federated learning	10
2.2 Workshop 3: Horizontal FL	12
2.2.1 Ease of deployment	13
2.2.2 Resource allocation	14
2.2.3 Source code	14
2.3 Workshop 4: Vertical FL.....	14
2.3.1 Vertical federated learning on ALS data	14
2.3.2 Vertical federated learning on complex modalities	15
2.3.3 Code	15
2.4 Server infrastructure	16
2.5 Client infrastructure.....	16
3 Flower Next	16
3.1 Flower Next: Messaging API.....	17
3.2 Data management	19
3.3 Privacy and security	19
3.4 Future defence strategies	20
3.5 Scalability, monitoring, and FAIRness	20
4 Future work.....	21
5 Milestone 4 verification.....	22
References	23
Annexes.....	26

List of Tables

Table 1 Results of live federated learning experiment performed with geographically distributed clients, illustrating the impact of geographical distribution on the experiment runtime.....	13
--	----

List of Figures

Figure 1 Diagram showing the difference between horizontal and vertical federated learning.....	12
Figure 2 Example of a client-side function when using the messaging API.	18

List of Abbreviations

Abbreviation	Original version
AAU	Aalborg University
AI	Artificial Intelligence
ALS	Amyotrophic lateral sclerosis
API	Application Programming Interface
CLEF	Conference and Labs of the Evaluation Forum
CLI	Command line interface
CPU	Central Processing Unit
CNN	Convolutional Neural Networks
CT	Computed tomography
DAG	Directed Acyclic Graph
DP	Differential Privacy
EBRAINS	European Brain Research Infrastructure
EEG	Electroencephalogram
EU	European Union
EuroHPC	European High-Performance Computing Joint Undertaking
FeTS	Federated Tumour Segmentation
FL	Federated Learning
Flower	A Federated Learning Framework
FOSDAM	Free and Open Source Software Developers' European Meeting
Gb	Gigabit
GB	Gigabyte
GDPR	General Data Protection Regulation
GiB	Gibibyte
GPU	Graphics Processing Unit
GPFS	General Parallel File System
gRPC	Cross-platform high-performance remote procedure call
HE	Homomorphic Encryption
HES-SO	University of Applied Sciences and Arts Western Switzerland
HPC	High-Performance Computing
JSON-RPC	JavaScript Object Notation Remote Procedure Call
k8s	Kubernetes
LLNL	Lawrence Livermore National Laboratory
ML	Machine Learning

Abbreviation	Original version
MLflow	Machine Learning Workflow (Open-source platform for ML lifecycle)
MNIST	Modified National Institute of Standards and Technology (dataset)
MRI	Magnetic Resonance Imaging
OIDC	OpenID Connect
PCA	Principal component analysis
PySyft	A Python Library for Secure and Private Deep Learning
RAM	Random Access Memory
REST	Representational State Transfer
RUMC	Radboud University Medical Centre
SLURM	Simple Linux Utility for Resource Management
SMPC	Secure Multiparty Computation
SURF	Dutch National Supercomputing Centre
TEE	Trusted Execution Environments
TLS	Transport Layer Security
UNIPD	University of Padua (Università degli Studi di Padova)
UNITO	University of Turin (Università degli Studi di Torino)
UCD	University of Colorado Denver
UI	User Interface
US	United States
vCPU	Virtual Central Processing Unit
VUB	Vrije Universiteit Brussel
W&B	Weights & Biases (Machine Learning Experiment Tracking)
XGBoost	eXtreme Gradient Boosting, a machine learning algorithm

1 Introduction

One of the primary objectives of the HEREDITARY project is to develop a federated learning and analytics infrastructure that functions within secure computing environments, including supercomputers. This configuration allows for distributed machine learning across multiple clients while prioritising data privacy and maintaining model security. By keeping data local, the system avoids centralisation, which is particularly crucial when handling sensitive medical data.

This deliverable reports on the work carried out so far in Task 2.6 during the first project period (Months 1-18). The task will continue until the end of Month 42. The current version focuses on the design of the federated infrastructure and its deployment within the scope of the project.

We tested the infrastructure in two scenarios with increasing complexity. The first focused on horizontal federated learning, mainly targeting Use Case 1. The second explored vertical learning, aligned with Use Case 2. Both use private ALS clinical data, with details available in Annex 4 ALS data description. Other public datasets are described in Section 2.3 Workshop 4: Vertical FL. These tests involved several participants, allowing us to verify communication protocols and examine the local hardware environments involved.

This deliverable confirms the completion of Milestone 6 and sets the foundation for the remaining work in this task. Future steps will continue to develop the current infrastructure, improving and expanding its features over upcoming reporting periods. It also further clarifies Milestone 4, verified by D2.14, by confirming the local computing infrastructure of UNITO, which has been refined and strengthened since the start of the project. Additional details are provided in Annex 3. The described work involved many members of the consortium. Data and requirements were supplied by partners: UNIPD (where the Department of Neuroscience provided the clinical data and requirements), RUMC, UNITO, and UCD. Technical contributions were made by partners working on WP3. UNIPD focused on integrating the polystore system, while AAU concentrated on privacy aspects, all coordinated to align with the federated learning approach.

1.1 Objectives of Task 2.6

The primary objective of Task 2.6 is to implement a federated learning infrastructure that is efficient, scalable, and capable of maintaining the privacy and security of data and models across a wide range of clients. To achieve this, in this deliverable, we describe the steps that were taken to design and test the infrastructure components:

- We address the need for data preprocessing, storage, and retrieval while adhering to privacy standards through code, dataset access, and some data hygiene practices see Section 3.2 Data management, Section 3.3 Privacy and security and Section 3.5 Scalability, monitoring, and FAIRness
- Model management is meant to facilitate efficient, reproducible training, validation, and deployment of machine learning models. In this deliverable, we describe an experiment orchestration using the Flower framework (Beutel2020; FlowerLabs, n.d.), the client-server setup, and the use of containerization and embedding, see Section 2.2.1 Ease of deployment, Section 3 (Flower Next overview), and Section 3.1 Messaging API & CLI.

- We planned to establish communication through secure interactions between clients and the central server, utilising technologies such as gRPC to protect data and model transmissions (Beutel2020). This deliverable provides in-depth coverage of horizontal versus vertical federated learning, the messaging API, and networking requirements. See Section 2.1, Explaining horizontal and vertical FL; Section 2.4, Server Infrastructure; Section 3.1, Messaging API; and Section 3.3, Privacy & Security.
- We address resource management concerns, such as allocation, load balancing, and fault tolerance, through a discussion on SLURM, Azure, and manual scheduling across partners (see Section 2.2.2, Resource Allocation, and the complementary details in Sections 2.4, Server Infrastructure, and 2.5, Client Infrastructure).

2 Workshops on horizontal and vertical FL

The design and development of the federated infrastructure have been conducted iteratively and collaboratively, involving all interested partners throughout the process. Two key moments in the design phase should be highlighted: Workshop 3, organised by SURF in Amsterdam on 30 and 31 January 2025, and Workshop 4, organised by SURF online on 19 May 2025. In these workshops, we focused on horizontal and vertical federated learning, creating codebases relevant to the project. We demonstrated that the code can be successfully executed between project participants within a federated setting. Given the importance of the developed code bases to the project, they can be quickly adapted to accommodate upcoming experiments.

2.1 Explaining horizontal and vertical federated learning

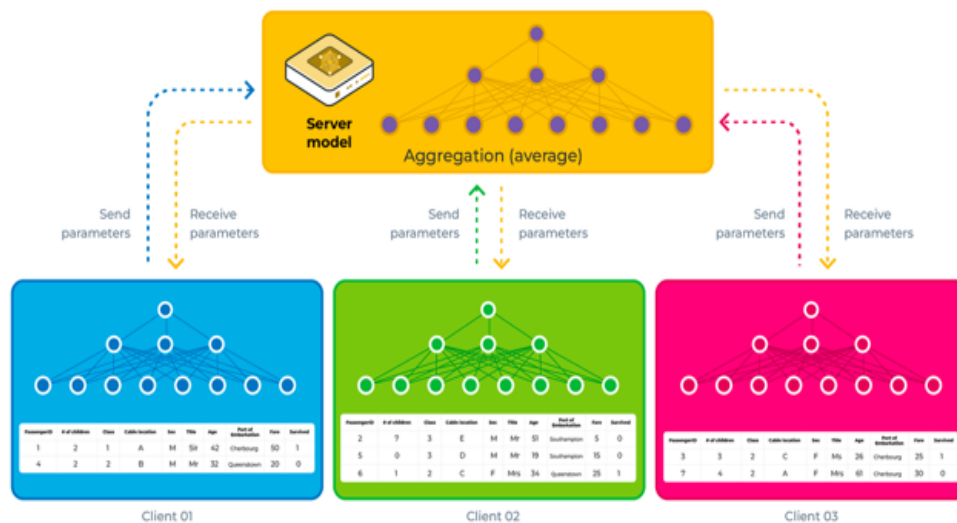
Commonly, federated learning is utilised in a horizontal federated learning setting, which means that all clients have the same type of data and locally have access to the labels for this data. For example, all clients may have undergone MRI scans of the brain and received associated segmentation labels. A lot of this data might be available in multiple centers, but due to, for example, privacy concerns, it can be hard or impossible to collect this data in a central location to train a model. In horizontal federated learning, a model is trained locally on all clients, and its weights are then communicated to the server. The server then aggregates all updates and combines the weights into a new model that is again distributed to the clients for further optimisation.

In the case of vertical federated learning, all clients have different or partially overlapping data, including duplicate data points. For example, client one might have an MRI scan of a patient, client two might have CT scans of a patient, and the server has access to the label for each patient in the dataset. In this case, the clients locally optimise a model to create an embedding of their local modality, which is then shared with the server. The server can then combine all embeddings and make a final prediction. In a neural network setting, we can then perform backpropagation to improve the network, but to improve the client networks, it is necessary to communicate the gradients to the clients. This means that for every individual optimisation step, server–client communication is required. In contrast, in the horizontal federated learning setting, a client can train on hundreds or thousands of examples before communicating with the server, as the client has access to both the data and the labels.

To summarise, vertical federated learning addresses problems where a data point is spread across multiple locations. In contrast, horizontal federated learning applies to situations where multiple locations have similar data points. In Figure 1, we display a visual comparison between horizontal and vertical federated learning, further illustrating the difference in model setup for both methods.

The frequent communication requirement of vertical federated learning significantly impacts training speed, especially when clients and servers experience high latency due to network limitations or geographical distance. In contrast, horizontal federated learning experiments are less affected by high communication because they require fewer communication rounds compared to the computational effort on clients. Nevertheless, for vertical federated learning, the communication cost can be prohibitive and needs further optimisation. Part of this high cost stems from clients being re-initialised with each server call, which can be mitigated by efficiently initialising models and other client-side components.

Horizontal Federated Learning



Vertical Federated Learning

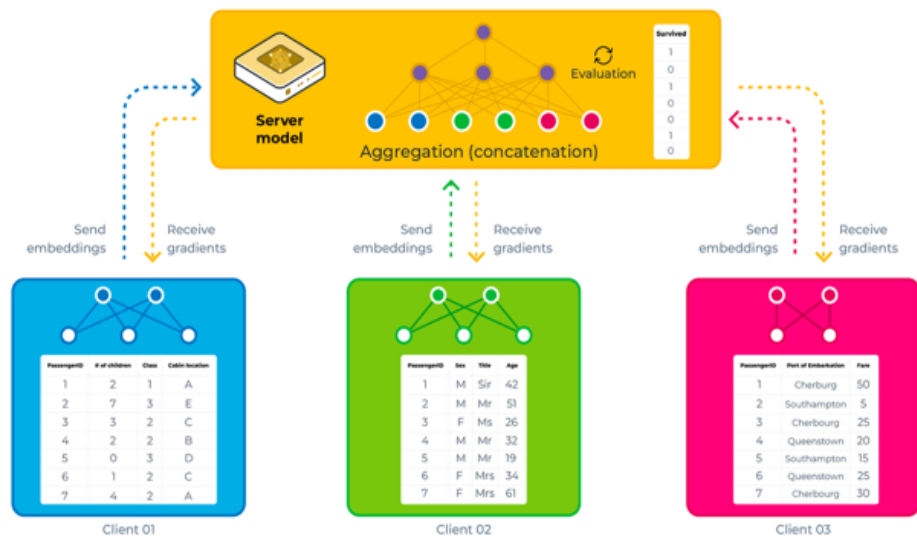


Figure 1 Diagram showing the difference between horizontal and vertical federated learning¹

2.2 Workshop 3: Horizontal FL

On 30 and 31 January 2025, an internal project workshop was held at the SURF office in Amsterdam, attended by 14 participants. The full agenda, list of participants, and notes can be found in Annex 1. During this workshop, the following sessions were organised:

First day

- SURF presented a session on the difference between how to organise experiments in the old version of the Flower framework and the new Flower Next framework.
- SURF presented a federated XGBoost approach using Dataset C of the ALS data, employing horizontal federated learning, which demonstrates the application of federated learning in Use Case 1: Neurodegenerative diseases phenotyping and prognosis evaluation. More details regarding the federated XGBoost model or the data preprocessing can be found in Annex 1, section 4.1.
- SURF presented a method to train the vision transformer-based Swin Unet3D model (Cai, 2023) to perform tumour segmentation on brain MRIs from the FeTS (Pati, 2021) dataset, using Flower with multiple clients, utilising the dataset partitions as described in the dataset. Notes on how this model was adapted for federated learning and what issues were encountered during development are in Annex 1, Section 4.2.
- UCD presented federated learning for Ophthalmology, a presentation demonstrating how to train a CNN classifier on images of the eye to classify glaucoma.

Second day

- SURF presented a method for performing spectral clustering in a federated fashion on Dataset C of the ALS data.
- A discussion was held on how to organise federated learning experiments, as it requires hardware and potential IT support on-site at multiple locations.

Following up on workshop 3, an additional virtual meeting has been organised with SURF, UNIPD, UNITO and UCD to investigate the impact of geographically distributed clients on the runtime of a federated learning experiment. Table 1 presents the time required for five rounds of horizontal federated learning when training an XGBoost model on tabular ALS data from Dataset C in various client configurations, which is the code base used in Workshop 3.

The simulation is highly efficient compared to running the distributed component on the same device. This difference stems from how jobs are allocated between the server and the client; in the distributed setup, clients are re-initialised each round, which adds extra time. The experiment used in Table 1 is very computationally efficient; therefore, initialisation and communication delays become noticeable. In row 3 of Table 1, we observe a very slight increase in runtime when distributing the experiment across multiple devices, but since all involved devices are physically in the same datacenter,

¹ <https://flower.ai/docs/examples/vertical-fl.html>

with a low latency connection (around 1 ms), the impact is minimal compared to running the experiment on a single laptop.

When involving a device in Italy, we observe approximately a 17% increase in runtime due to the additional latency between the devices. Then, when an extra device from the US is added to the experiment, the runtime increases by 70% compared to the experiment involving only devices at SURF datacentres. A 70% rise in execution time can significantly affect an experiment; however, it should be noted that this specific experiment is computationally inexpensive, meaning that communication represents a large portion of the total runtime. For more computationally intensive experiments, where clients might perform calculations for 10 or 20 minutes before communicating with the server, the impact of communication on the overall runtime is significantly lower.

Server	Participants	Time for five rounds
Local simulation (MacBook Pro M3)	2 clients, in local simulation (2x MacBook M3)	9.83 seconds
Local flower superlink (MacBook Pro M3)	Local flower superlink (MacBook Pro M3)	51.16 seconds
SURF Research Cloud	2 x SURF Snellius supercomputer nodes	52.54 seconds
SURF Research Cloud	UNIPD + SURF Snellius	61.51 seconds
SURF Research Cloud	UNIPD + laptop in Italy	58.52 seconds
SURF Research Cloud	UNIPD + UNITO	67.51 seconds
SURF Research Cloud	UNIPD + UCD ²	94.56 seconds
SURF Research Cloud	UNIPD + UCD + SURF Snellius	89.37 seconds
SURF Research Cloud	UNIPD + UNITO + SURF Snellius	68.64 seconds

Table 1 Results of live federated learning experiment performed with geographically distributed clients, illustrating the impact of geographical distribution on the experiment runtime. One round consists of the server instructing the clients to perform a task, the clients performing this task, the clients returning the results, and the server aggregating the results.

2.2.1 Ease of deployment

During workshop 3, it was concluded that the local setup should be as simple as possible to reduce the impact of technical issues on the experiments. To achieve this, the XGBoost method presented in Workshop 3 has been implemented in Docker, allowing both the server and clients to run within the same Docker image. The Docker image was based on the Ubuntu Noble Numbat image and used pip to install the requirements defined by the project. Project dependencies are defined in the “pyproject.toml” file,

² The UCD server was hosted on Azure in the “South Central US (Zone 2)” region, located in Texas.

which is created automatically when a new Flower project is initialised and contains all relevant dependencies. Since the Docker definition depends only on the “pyproject.toml”, it can be easily adapted to other projects.

2.2.2 Resource allocation

During Workshop 3, one of the sessions organised by SURF discussed the resource allocation problem. To start an experiment, hardware must be available to all participants. Different project partners use various scheduling strategies; some utilise SLURM, others rely on commercial cloud platforms like Azure, while some lack scheduling capabilities or organise scheduling via text messages within a group. This last approach presents a significant obstacle to efficient resource allocation, as it cannot be automated. The interim solution has been to regularly organise meetings during experiments, where everyone is online and monitors their hardware. We are now exploring a solution that provides a single, policy-driven REST/gRPC interface while allowing each partner to maintain or retire their local scheduler at their own pace. Several projects have addressed this topic, including UNICORE 10 (UNICORE2025), Hydra broker (Alsaadi2024), and Flux framework/operator (Ahn2020). The UNICORE service layer is a mature, EuroHPC-proven software suite that exposes jobs, data, and workflows through stable REST endpoints; it already facilitates EBRAINS users switching between the five Fenix Tier-1 systems, and the latest 10.2.x release introduces first-class OIDC token flows for detailed access control. The Hydra broker is a lightweight JSON-RPC broker capable of co-allocating cloud VMs and on-prem HPC nodes within the same DAG; benchmarks from 2024 in the reference paper demonstrate linear strong and weak scaling up to approximately 10,000 concurrent tasks, and we can leverage the Tier-2 “Hydra” cluster at VUB for EU-hosted testing. The Flux nested scheduler can run within Kubernetes, under Slurm, or as a standalone micro-cluster. LLNL’s tests report significantly reduced job start-up latency for large task swarms, and the open-source Flux Operator makes scaling on any Kubernetes service straightforward. We will detail this in D2.15, scheduled for Month 22.

2.2.3 Source code

All code used in Workshop 3 can be found on GitHub here: https://github.com/sara-nl/Hereditary/tree/third_workshop/

2.3 Workshop 4: Vertical FL

On May 19 2025, SURF presented two methods for performing vertical federated learning. The first method was developed for ALS data, the second method was developed for the multimodal BrainLat dataset (Prado2023), exploring how multiple complex modalities, such as EEG and MRI, can be combined efficiently through vertical federated learning. More details regarding the experiments and participants in the workshop can be found in Annex 2.

2.3.1 Vertical federated learning on ALS data

The ALS data of Dataset C can be split into two partitions: the first consisting of patient demographic data, and the second consisting of (blood)test and questionnaire data. This code was demonstrated to run geographically distributed with the server running in the SURF Research Cloud, one client running on SURF’s Snellius supercomputer, and a final client running at UNIPD. Docker images had been prepared for workshop 4, to

reduce the setup burden for participants, which has dramatically increased the ease of deployment and ensures consistency between clients, thereby also improving the reproducibility of the experiments.

For most modalities, a neural network can provide an embedding, allowing for the adaptation of the existing codebase to work with different modalities with minimal adjustments to the vertical federated learning infrastructure. In this case, we simulated use-case 2 of the project using the ALS dataset (Faggioli2024, Faggioli2023a, Faggioli2023b), where we perform vertical federated learning with two modalities.

2.3.2 Vertical federated learning on complex modalities

We have acquired the publicly available BrainLat (Prado2023) dataset, which comprises ~760 patients and contains multiple types of MRIs, EEGs, and tabular data from several sites, making it a suitable candidate for simulating a multimodal, multi-site vertical federated learning scenario.

In our initial experiments, we used this MRI and EEG data to develop a comprehensive framework that adapts SigLIP (Zhai2023), a contrastive learning method based on the multi-modal CLIP (Radford2021) model, for learning from federated, multimodal medical data under strict privacy constraints. This approach allows the alignment of diverse data types, such as EEG, MRI, clinical text, and genomics, into a shared embedding space, even when data is scattered across different institutions. The framework works in three phases: (1) *Pretraining*, where modality-specific encoders are trained on publicly available datasets using SigLIP; (2) *Federated learning*, where these encoders are distributed to individual clients for local fine-tuning — each client may have different modalities and data amounts — followed by central aggregation of updates during communication rounds; and (3) *Posttraining*, an optional centralised fine-tuning phase to improve the shared representations. To handle varied modality availability, the framework supports strategies such as assigning specialised encoders to each client or requiring all clients to share at least one standard modality (e.g., text) to allow unified updates. In the BrainLat dataset, we used a 3D convolutional encoder for MRI and a transformer-based encoder, similar to LLaMA-3 (Grattafiori2024), for EEG, both mapped into a shared embedding space. This setup shows the potential of SigLIP to enable privacy-preserving, collaborative learning from isolated, multimodal medical datasets.

Unfortunately, there are realistically only two out of four MRI modalities (T1 weighted and diffusion weighted) within the BrainLat dataset that contain sufficient data for deep learning experiments in BrainLat, and only twenty patients have both MRI and EEG data available. However, the method can still be helpful given the many complex modalities for which data is available in the project.

2.3.3 Code

All code used in the fourth workshop can be found on GitHub here: https://github.com/sara-nl/Hereditary/tree/fourth_workshop/

2.4 Server infrastructure

During the experiments in workshops 3 and 4, the Flower server has been running on an instance on the SURF Research Cloud environment³. The instance had access to 16 CPU threads and 64 GB of RAM. Generally, this would be more than sufficient for experiments. For horizontal federated learning, averaging strategies are often not significantly computationally expensive and in the case of vertical federated learning, only a small fully connected network needs to be applied to embeddings, which is generally faster on a CPU than a GPU due to the overhead of transferring data between the CPU and GPU.

A server should have a high bandwidth (at least 1 GB) and a low-latency connection, because data needs to be transferred to the server frequently, and all other hardware is idle during the transfer. In the case of horizontal federated learning, we need to transfer model weights from each client to the central server. For example, a ResNet50 image classification model has 98 MB of weights when stored in FP32. In the case of vertical federated learning, embeddings need to be transferred to the server. For example, if the test set consists of 2000 samples, each embedded in 512 features, resulting in an 8MB file for transfer from each client.

Regarding networking, a firewall should not block incoming connections to the server, and the server requires a total of three open ports. By default, these are 9091, 9092 and 9093. Clients should be connected to the internet and do not require open ports.

2.5 Client infrastructure

The client infrastructure is strongly dependent on the data that needs to be processed, and in the case of deep neural networks, it can often benefit from having GPU access.

Partner infrastructure has been discussed in Deliverable 2.14 Annex 1, 3 and 4 (Podareanu2024). Although the arrangement of the UNITO infrastructure was still underway at the time of Deliverable 2.14's submission, the process has since been completed, and the details are provided in Annex 3. All partners have infrastructure that includes GPU's.

3 Flower Next

Since the start of the project, the Flower library has been utilised to implement federated learning experiments. In March 2024, Flower Next⁴ was released, which altered the way servers and clients operate and how users can conduct experiments.

In the previous Flower framework, a user would implement a server app and a client app script. The user would then start the server, followed by the clients which connected to the server. Once the experiment was completed, all scripts would shut down. This meant that there would only be a temporary link between all participants of the experiment.

The significant change in Flower Next is that the server and clients are now persistent, and users can submit experiments to the server, which are then distributed to the clients.

³ <https://servicedesk.surf.nl/wiki/spaces/WIKI/pages/9798172/SURF+Research+Cloud>

⁴ <https://flower.ai/blog/2024-04-03-announcing-flower-1.8-release>

As long as the server and clients have the required dependencies installed, any experiment can run.

This comes with both advantages and disadvantages.

Advantages:

- A persistent server allows users to interact with the server, even if they do not own it, and enables them to inspect logs of submitted runs.
- Flower Next comes with a CLI that allows users with credentials to connect to the server, submit experiments, stop experiments, and read log files. Multiple users can utilise the network of servers and clients independently and submit experiments without requiring local support for each experiment. When submitting multiple experiments, the experiments will be executed sequentially.

Disadvantages:

- Any code submitted to the server will be distributed to the clients and executed, which could be malicious. One of the project members at SURF has attended the Flower Summit 2025 where a presentation⁵ was given by the Flower developer team on the Flower platform, which included a demonstration of the Flower Superlink UI, which is a web-based interface through which admins and federation members can log in, see results and inspect connected devices. This interface could easily be expanded to include an approval process for each submitted experiment, thereby addressing this disadvantage.
- It is technically more complicated to select a subset of the clients that are connected to the server, if that is required for a specific experiment.
- Code from the old version of Flower is, at this point, still compatible with the current Flower library, but will soon be deprecated. Porting between the two frameworks is not always trivial, for example, when hydra configuration files are used, which need to be loaded when a program is started. When using Flower Next, these configuration files are not loaded correctly; thus, a major refactor of the code is needed to maintain all functionalities provided by the configuration. This potentially reduces the usability of legacy code in examples or code published with papers, but Hereditary itself does not use any code impacted by this change.

3.1 Flower Next: Messaging API

Flower Next has followed the traditional flower setup, defining a server and a client, each with a set of functions that are called throughout the training process. This is convenient for many projects that train a conventional model on their private data, as essentially only data loading and preprocessing code has to be created. This is the procedure that is followed in the traditional server/client app setup:

1. Initialisation phase
 - Strategy requests initial model parameters with `initialize_parameters()`
 - The server returns parameters to the Strategy
2. Federated training phase

⁵ <https://www.youtube.com/watch?v=mwm0yPhFdbM>

- Strategy configures training with `configure_fit()`
- The server receives a list of client proxies and training configurations
- The server distributes training instructions to clients
- Clients perform training and return `FitRes` results
- Strategy aggregates results with `aggregate_fit()`
- Aggregated model parameters are returned to the Strategy
- 3. Centralised evaluation phase
 - Strategy evaluates the model centrally with `evaluate()`
 - Centralised evaluation results are returned
- 4. Federated evaluation phase
 - Strategy configures distributed evaluation with `configure_evaluate()`
 - The server receives a list of client proxies and evaluation configurations
 - The server distributes evaluation instructions to clients
 - Clients perform evaluation and return `EvaluateRes` results
 - Strategy aggregates results with `aggregate_evaluate()`
 - Aggregated evaluation results are returned to Strategy
- 5. Next round
 - The process continues to the next round of federated training

This represents a complete round of federated learning where model parameters are distributed, locally trained, evaluated, and aggregated in a coordinated manner.

However, some use cases require more specific client-server interaction that does not fit in the regular train cycle of Flower. For this case, Flower provides the Messaging API. When using the messaging API, functions are defined that can be called by the server. On the server, a user can define any procedure.

```
@app.query("set_mean_std")
def set_mean_std(message: Message, context: Context):
    """Set the mean and std of the data."""
    client_state.global_mean = message.content["config"]["mean"]
    client_state.global_std = message.content["config"]["std"]
    reply_record = RecordDict({"result": ConfigRecord({"success": True})})
    return Message(reply_record, reply_to=message)
```

Figure 2 Example of a client-side function when using the messaging API.

In Fig. 2, we demonstrate an example of a client-side function when using the messaging API. The server can call the “set_mean_std” function with a specific message at any point in time, and the function decorated with the “set_mean_std” query will be executed. This allows for larger freedom in the federated process and enables federated analytics but requires more custom code for processes that are currently supported by strategies in the Flower Framework, such as horizontal federated learning. Furthermore, the option to dynamically call different functions from different clients opens the door to easily customising a task per client based on, for example, the available compute. When one client has a fast GPU, and another client has a slower GPU and the two clients are requested to perform the same task, the faster GPU will spend time idling. This can be prevented by, for example, adjusting the batch size, adjusting the model's precision, or selectively updating parameters. Dynamically adjusting the workload per client can also

be performed through the initialisation function when using a Flower Strategy, but is significantly easier using the messaging API.

3.2 Data management

In the federated learning space, it is common for each client participating in the experiment to have a dataset of a shared modality. However, these datasets may differ slightly due to distribution shifts, varying measurement methods (e.g., different scanners for histopathology or MRI data), and may also differ in practical storage implementations.

All on-premise data stores use LUKS full-disk encryption. Keys are rotated yearly and stored in an HSM; access tokens are issued via the OIDC layer already deployed for Flower. The consent ID recorded in the FAIR metadata governs whether the data can be used for (a) model training only or (b) training + derivative sharing; this flag is enforced by the server when it assembles the client list for a run.

In Flower, a client can be started with specific persistent parameters, which can be used to identify the client. Using such a parameter, we can utilise client-specific data loading throughout the client code, allowing the experiments to utilise data of different types, different folder structures and compensate for local distribution changes in the data.

3.3 Privacy and security

Flower leverages secure aggregation protocols SecAgg and SecAgg+, via its [SecAggWorkflow](#), ensuring each client submits its masked update so the server can only see the aggregated result. This protects against eavesdropping or model inversion attempts and is inherently tolerant of client dropouts up to a configurable threshold. In terms of computational costs, the lightweight secret sharing and mask operations add minimal CPU overhead. These aggregation protocols introduce an additional round of communication during setup and unmasking, typically 1–2 times the time required for gradient transmission. Against baseline horizontal training (a few aggregate rounds), secure aggregation increases runtime by ~10–30%, but significantly improves privacy.

In our Workshop 3 horizontal XGBoost experiments, adding SecAgg is expected to increase the runtime from ~52 seconds to 60–68 seconds. For the resource-limited vertical setup in Workshop 4, the overhead is proportionally smaller but comparable relative to the overall training time.

All gRPC communication is encrypted via TLS, using channel and call credentials to secure each RPC. This prevents man-in-the-middle attacks and ensures data integrity in transit. TLS typically adds <5% latency and CPU cost vs. unencrypted gRPC, negligible relative to secure aggregation's overhead.

In the introduction of Section 3, we emphasise that Flower does not permit client owners to approve code before it is executed. This issue can be addressed in multiple ways, such as through the management UI mentioned earlier, provided by the Flower framework. Alternatively, there is an additional framework, Vantage6 (Moncada, 2021; Smits, 2022), which enables server and client workflows to be prepared in Docker images and can be utilised. All client owners then approve these images before conducting the experiment. The Flower server and client can be configured in Docker

and operated via Vantage6 to leverage Flower's advanced federated learning capabilities alongside Vantage6's approval system.

3.4 Future defence strategies

For Deliverable 2.15, we plan to integrate additional privacy protections:

1. Differential Privacy (DP) – Both local and central DP pipelines will be introduced, clipping individual updates and adding calibrated noise to satisfy ϵ -DP guarantees.
2. Trusted Execution Environments (TEEs) – Consider Intel SGX or similar technologies for the secure processing of embeddings and gradients as an alternative to homomorphic encryption.
3. Anomaly detection & poisoning defences – Deploy statistical checks or gradient-view filters to detect and isolate malicious client agents in future iterations.

3.5 Scalability, monitoring, and FAIRness

Flower Next's persistent architecture supports large-scale deployment. Core mechanisms include:

- Job queueing and CLI-driven experiment dispatch
- Messaging API enabling dynamic batch sizes and workload tuning per client
- Gradient compression (e.g. quantisation, PCA) and embedding pre-caching to mitigate the high overhead of vertical federated learning

When implementing these new scalability features, horizontal runs with moderate compression see ~10–20% speedups. Vertical runs, where communication is the dominant factor, may experience acceleration of 30–50% compared to raw streaming, depending on the compression ratio.

Flower Next also automatically captures key metrics, such as round duration, loss, accuracy, and client availability, through its command-line interface (CLI) and server logs. This enables centralised monitoring for reproducibility, debugging, and performance tracking.

We would like to integrate with frameworks like MLflow or Weights & Biases in the future for richer dashboards and round-by-round insights.

We're exploring client ownership and incentive fairness using Shapley value-based approaches(Song2019). While early Shapley implementations incur significant computational costs, recent gradient-reconstruction variants (Xu2021) offer on-the-fly assessments that are compatible with real-time federated learning (FL) operations. Full Shapley can add 2–5 times per round compute; gradient-based approximations increase run time <30% but yield reliable fairness estimates suitable for cross-silo setups (Zheng2023).

4 Future work

To fully complete the federated learning infrastructure beyond its current state, several key areas require further investigation and development. Progress on this work will be reported in Deliverable 2.15.

First, while encrypted gRPC/TLS communication and secure aggregation (SecAgg/SecAgg+) provide strong privacy guarantees, a formal threat model is still missing. We plan to define adversarial settings, such as honest-but-curious and Byzantine clients, and investigate defences like anomaly detection and malicious aggregation to harden the system.

Second, our current setup does not yet address the challenges of client heterogeneity and fairness. There is no dynamic client weighting, fairness-aware participant selection, or support for model personalisation. We will explore strategies such as adaptive sampling, personalised federated learning, and fairness-aware aggregation to ensure distributed clients contribute equitably and to mitigate bias arising from non-IID data distributions.

Third, while basic compression (e.g., PCA, quantisation) has been applied, the infrastructure lacks hierarchical aggregation and federated transfer learning mechanisms necessary for large-scale, multi-site deployments. Drawing on hierarchical federated learning frameworks, we will pilot multi-tier aggregation architectures and model transfer strategies to improve bandwidth efficiency and scalability.

Fourth, we have a planned integration of Differential Privacy (DP) and Trusted Execution Environments (TEEs). To balance privacy and utility, we will run pilot experiments to quantify accuracy and runtime impacts across ϵ -DP budgets, and evaluate performance using hierarchical DP techniques such as SDP. Currently, there is a Ph.D. student working on DP for different types of data. One of the biggest problems in DP is to balance data privacy and utility, and most of the existing methods only focus on tabular data, making them not suitable for medical images or genomic data. The problem is bigger when considering multimodal data, such as medical images paired with tabular data, as the differential privacy methods should consider the relation between both types of data.

Fifth, monitoring capabilities are currently limited to CLI logs. We will integrate tools like MLflow and Weights & Biases for round-by-round metrics, implement fairness dashboards, and set up automated alerting for client dropouts or anomalies, complementing persistent monitoring facilitated by Flower Next.

Sixth, with the introduction of the Flower Next, any submitted code may run on clients without restriction. We will thus design a sandboxed code review process to inspect and vet client-side code before deployment, thereby preventing the execution of malicious or faulty code.

Seventh, our work on the BrainLat dataset was limited by the low overlap in multimodal patient data and data preprocessing errors. To address this, we plan to explore synthetic data augmentation and seek alternative multimodal datasets to validate vertical federated workflows more robustly.

Eight, no standardised evaluation metrics or baselines are currently in place, particularly for comparing horizontal versus vertical FL performance. We will define metrics such as

convergence rates, robustness, fairness, and communication efficiency, and establish reference benchmarks across configurations.

Nine, we are studying non-IID and their impact on federated learning. Non-IID presents as data in different institutions may have different distributions as a result the trained model might have an unfair performance across institutions. For instance, an institution with more data might unbalance the model to its distribution. We are studying how to mitigate this effect by sampling techniques of other manners to equalize the data.

Finally, though GDPR and encrypted communication are implemented, formal governance mechanisms—including data sovereignty, audit trail infrastructure, and automated consent management—are not yet in place. We aim to implement provenance tracking, consent workflows, and audit logging to ensure compliance and accountability.

5 Milestone 4 verification

In Deliverable 2.14, it was reported that UNITO was still working on completing the infrastructure setup for federated learning. In Section 2.2, where we reported on Workshop 3, we included Table 1 with experiment execution times. UNITO also took part in these experiments utilizing the compute infrastructure that they intend to use for Hereditary, their definitive hardware specification has been included in Annex 3. Hardware specifications of other partners can be found in Deliverable 2.14, Annex 1, 3 and 4 (Podareanu2024). As UNITO participated successfully in experiments with SURF, UNIPD and UCD utilizing hardware intended for HEREDITARY, the milestone has now been verified further.

References

The bibliographic entries are arranged in lexicographical order based on the key, following the APA style. This enables us to place the entries and citations in the table and text in any sequence, allowing for later sorting while ensuring consistency.

Key	Reference
Ahn2020	Ahn, D. H., Bass, N., Chu, A., et al. (2020). <i>Flux: Overcoming scheduling challenges for exascale workflows</i> . <i>Future Generation Computer Systems</i> , 110, 202–213.
Alsaadi2024	Alsaadi, A., Turilli, M., & Jha, S. (2024, September). <i>Hydra: Brokering cloud and HPC resources to support the execution of heterogeneous workloads at scale</i> . In <i>Proceedings of the 14th Workshop on AI and Scientific Computing at Scale using Flexible Computing Infrastructures (FlexScience'24)</i> . ACM. https://doi.org/10.1145/3659995.3660040
Beutel2020	Beutel, D., Topal, T., Mathur, A., Qiu, X., Parcollet, T., & Zhao, Y. (2020). <i>Flower: A friendly federated learning research framework</i> . <i>Proceedings of the 2020 IEEE International Conference on Big Data (Big Data)</i> , 1001–1010. https://doi.org/10.1109/BigData50022.2020.9377766
Bonawitz2017	Bonawitz, K., Ivanov, V., Kreuter, B., Marcedone, A., McMahan, H. B., Patel, S., & Song, S. (2017). Practical secure aggregation for federated learning on user-held data. <i>Proceedings of the 2017 ACM SIGSAC Conference on Computer and Communications Security (CCS '17)</i> , 1175–1191. https://doi.org/10.1145/3133956.3133982
Cai2023	Cai, Y., Long, Y., Han, Z. et al. Swin Unet3D: a three-dimensional medical image segmentation network combining vision transformer and convolution. <i>BMC Med Inform Decis Mak</i> 23, 33 (2023). https://doi.org/10.1186/s12911-023-02129-z
Cheon2017	Cheon, J. H., Kim, A., Kim, M., & Song, Y. (2017). Homomorphic encryption for arithmetic of approximate numbers. In T. Takagi & T. Peyrin (Eds.), <i>Advances in Cryptology—ASIACRYPT 2017</i> (pp. 409–437). Springer. https://doi.org/10.1007/978-3-319-70694-8_15
Faggioli2023a	Faggioli, G., Guazzo, A., Marchesin, S., Menotti, L., Trescato, I., Aidos, H., Bergamaschi, R., Birolo, G., Cavalla, P., Chiò, A., Dagliati, A., de Carvalho, M., Di Nunzio, G. M., Fariselli, P., García Dominguez, J. M., Gromicho, M., Longato, E., Madeira, S. C., Manera, U., Silvello, G., Tavazzi, E., Tavazzi, E., Vettoretti, M., Di Camillo, B., & Ferro, N. (2023). Intelligent disease progression prediction: Overview of iDPP@CLEF 2023. In <i>Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the Thirteenth International Conference of the CLEF Association (CLEF 2023) (Lecture Notes in Computer Science, Vol. 14163)</i> . Springer.
Faggioli2023b	Faggioli, G., Guazzo, A., Marchesin, S., Menotti, L., Trescato, I., Aidos, H., Bergamaschi, R., Birolo, G., Cavalla, P., Chiò, A., Dagliati, A., de Carvalho, M., Di Nunzio, G. M., Fariselli, P., García Dominguez, J. M., Gromicho, M., Longato, E., Madeira, S. C., Manera, U., Silvello, G., Tavazzi, E., Tavazzi, E., Vettoretti, M., Di Camillo, B., & Ferro, N. (2023). Overview of iDPP@CLEF 2023:

Key	Reference
	The Intelligent Disease Progression Prediction Challenge. CLEF (Working Notes) 2023, CEUR Workshop Proceedings, 3497, 1123–1164.
Faggioli2024	Faggioli, G., Menotti, L., Marchesin, S., Chiò, A., Dagliati, A., de Carvalho, M., Gromicho, M., Manera, U., Tavazzi, E., Di Nunzio, G. M., Silvello, G., & Ferro, N. (2024). An extensible and unifying approach to retrospective clinical data modeling: The BrainTeaser ontology. <i>Journal of Biomedical Semantics</i> , 15(16). https://doi.org/10.1186/s13326-024-00317-y
FlowerLabs,n.d.	Flower Labs. (n.d.). <i>Flower: A Friendly Federated AI Framework</i> . https://flower.ai
Grattafiori2024	Grattafiori, A., Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., ... & Vasic, P. (2024). The llama 3 herd of models. <i>arXiv preprint arXiv:2407.21783</i> . https://doi.org/10.48550/arXiv.2407.21783
Moncada2021	Moncada-Torres, A., Martin, F., Sieswerda, M., Van Soest, J., & Geleijnse, G. (2021, January). VANTAGE6: an open source priVAcY preserviNg federaTeD leArninG infrastructurE for Secure Insight eXchange. In <i>AMIA annual symposium proceedings</i> (Vol. 2020, p. 870).
Paillier1997	Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In J. Stern (Ed.), <i>Advances in Cryptology—EUROCRYPT '99</i> (pp. 223–238). Springer. https://doi.org/10.1007/3-540-48910-X_16
Pati2021	Pati, S., Baid, U., Zenk, M., Edwards, B., Sheller, M., Reina, G. A., ... & Bakas, S. (2021). The federated tumor segmentation (fets) challenge. https://doi.org/10.48550/arXiv.2105.05874
Podareanu2024	Podareanu, D., Dell'Aglio, D., Romanovych, A., & Silvello, G. (2024). Deliverable 2.14: Computing infrastructures. <i>Zenodo</i> . https://doi.org/10.5281/zenodo.14034331
Prado2023	Prado, P., Medel, V., Gonzalez-Gomez, R., Sainz-Ballesteros, A., Vidal, V., Santamaría-García, H., ... & Ibañez, A. (2023). The BrainLat project, a multimodal neuroimaging dataset of neurodegeneration from underrepresented backgrounds. <i>Scientific Data</i> , 10(1), 889. https://doi.org/10.1038/s41597-023-02806-8
Radford2021	Radford, A., Kim, J. W., Hallacy, C., Ramesh, A., Goh, G., Agarwal, S., ... & Sutskever, I. (2021, July). Learning transferable visual models from natural language supervision. In <i>International conference on machine learning</i> (pp. 8748–8763). PmLR. https://doi.org/10.48550/arXiv.2103.00020
Ryffel2018	Ryffel, T., Trask, A., Dahl, M., Wagner, B., Mancuso, J., Rueckert, D., & Passerat-Palmbach, J. (2018). A generic framework for privacy-preserving deep learning. <i>arXiv preprint arXiv:1811.04017</i> . https://arxiv.org/abs/1811.04017
Smits2022	Smits, D., Van Beusekom, B., Martin, F., Veen, L., Geleijnse, G., & Moncada-Torres, A. (2022). An improved infrastructure for privacy-preserving analysis of patient data. In <i>Advances in Informatics, Management and Technology in Healthcare</i> (pp. 144–147). IOS Press. https://doi.org/10.3233/shti220682

Key	Reference
Song2019	Song, T., Tong, Y., & Wei, S. (2019). Profit allocation for federated learning. 2021 IEEE International Conference on Big Data (Big Data), 2577–2586. https://doi.org/10.1109/bigdata47090.2019.9006327
UNICORE2025	UNICORE Forum e.V. (2025, June 18). <i>UNICORE: Distributed computing and data resources</i> (Version 10.2.1). https://www.unicore.eu/
Xu2021	Xu, X., Lyu, L., Ma, X., Miao, C., Foo, C. S., & Low, B. K. H. (2021). Gradient driven rewards to guarantee fairness in collaborative machine learning. <i>Neural Information Processing Systems</i> , 34. https://papers.nips.cc/paper/2021/hash/8682cc30db9c025ecd3fe433f8ab54c-Abstract.html
Zhai2023	Zhai, X., Mustafa, B., Kolesnikov, A., & Beyer, L. (2023). Sigmoid loss for language image pre-training. In <i>Proceedings of the IEEE/CVF international conference on computer vision</i> (pp. 11975-11986). https://doi.org/10.48550/arXiv.2303.15343
Zheng2023	Zheng, S., Cao, Y., & Yoshikawa, M. (2023). Secure shapley value for Cross-Silo federated Learning. <i>Proceedings of the VLDB Endowment</i> , 16(7), 1657–1670. https://doi.org/10.14778/3587136.3587141
Ziller2018	Ziller, M., Peters, A., & Wagner, M. (2018). Secure aggregation for federated learning using PySyft. <i>Proceedings of the 2018 International Conference on Privacy, Security, Risk and TruPASSAT'18</i> '18), 412-422. https://doi.org/10.1109/PASSAT.2018.153

Annexes

Number	Name
1	Workshop 3: Agenda, attendance and detailed explanation of presented work
2	Workshop 4: Agenda, attendance and detailed explanation of presented work
3	Hardware description UNITO
4	ALS data description