ModularFed: Leveraging Modularity in Federated Learning Frameworks

Mohamad Arafeh[a], Hadi Otrok[c], Hakima Ould-Slimane[b], Azzam Mourad[d,e], Chamseddine Talhi[a], Ernesto Damiani[c]

[a]*Department of Software Engineering, Ecole de Technologie Superieure (ETS), Montreal, QC, Canada*
[b]*Department of mathematics and computer science, Universite de Quebec a Trois-Rivieres (UQTR), Canada*
[c]*Center of Cyber-Physical Systems (C2PS), Department of EECS, Khalifa University, Abu Dhabi, UAE*
[d]*Cyber Security Systems and Applied AI Research Center, Department of CSM, Lebanese American University, Lebanon*
[e]*Division of Science, New York University, Abu Dhabi, United Arab Emirates*

Abstract

Numerous research recently proposed integrating Federated Learning (FL) to address the privacy concerns of using machine learning in privacy-sensitive firms. However, the standards of the available frameworks can no longer sustain the rapid advancement and hinder the integration of FL solutions, which can be prominent in advancing the field. In this paper, we propose ModularFed, a research-focused framework that addresses the complexity of FL implementations and the lack of adaptability and extendability in the available frameworks. We provide a comprehensive architecture that assists FL approaches through well-defined protocols covering three dominant FL paradigms: adaptable workflow, datasets distribution, and third-party application support. Within this architecture, protocols are blueprints that strictly define the framework's components' design, contribute to its flexibility, and strengthen its infrastructure. Further, our protocols aim to enable modularity in FL, supporting third-party plug-and-play architecture and dynamic simulators coupled with major built-in data distributors. Additionally, the framework support wrapping multiple approaches in a single environment to enable consistent replication of FL issues such as clients' deficiency, data distribution, and network latency, which

entails a fair comparison of techniques outlying FL technologies. In our evaluation, we examine the applicability of our framework addressing major FL domains, including statistical distribution and modular-based resource monitoring tools and client selection. Moreover, our comparison analysis indicates that our architecture has an inconsiderable impact on performance compared to other approaches.

Keywords: Federated Learning, Machine Learning, Non-IID, Privacy.

---

1. Introduction

Federated learning is a distributed technique that emerged from the need for a scheme to address the increasing restriction on users' data privacy. In FL, users train ML models locally and send them to an external server while keeping their raw data intact on their devices. Furthermore, the server, in turn, performs a particular aggregation algorithm, which is the core of any FL approach. Aggregation algorithms have the role of combining the received clients' models into one global model. Such a procedure takes place over multiple rounds and ends when the model is reliable enough to have accurate predictions. FL can be defined by a Global Model (GM), initialized by the server and holds the configuration and the initial parameter values. A client selection procedure, in which part of the client pool is selected either randomly or by specified constraints [1, 2, 3, 4], to collect the GM shared by the server and train it. Finally, an aggregation algorithm allows the server to combine the clients' updates and commit them to GM. Following such a scheme, external parties can benefit from users' collected data without violating their privacy.

Despite its promising rationale and the numerous recent investigations [5, 6, 7, 8], FL's trustworthiness and performance are majorly affected by statistical, security and resource concerns, restricting further development in the field. For instance, FL implies individual training from clients using their local datasets, which in most cases, is deemed biased toward the clients' behaviours and surroundings. Such a situation signifies a non-identical and independent (Non-IID)

data distribution in which data imbalance exists by either size or content (features/classes). Referring to [9], experiments with Non-IID constitute evidence of serious drawbacks on accuracy and loss developments. Approaches such as [10, 11, 12, 13] attempted to exploit the connection between the clients and their model parameters by clustering algorithms to separate them into distinct FL contexts. Having only similar clients working together can simulate an IID environment. On the other hand, the works presented in [14, 15, 16] take advantage of the aggregation procedure to fix the model shift caused by training on Non-IID clients. Another form of solution operates on the starting parameters. For instance, in [4, 17], the experiments show that starting from a pretrained model can boost the performance in a Non-IID environment. In terms of security, various defence mechanisms aim to protect against manipulation [18, 19, 20], trustworthiness [21, 22, 23], and inference attacks [24, 25]. Regarding resource consumption, development focuses on reducing the communication overhead as in [26, 27] or energy consumption as in [28, 29, 30, 31]. Other approaches in [3, 32] acted on the produced clients' model quality to reduce the communication rounds. Their experiments show a significant improvement in weight development through selection algorithms, increasing the convergence rate and reducing the total cost.

From the aforementioned approaches, we underline essential facts which are the motivation behind this work:

- The majority of the aforementioned approaches focus on three areas: aggregation, client selection, and communication [14, 15, 2, 32, 26, 27, 33] that can be targeted to create a flexible environment for future solutions.
- With few modifications, it is possible to integrate existing solutions from other domains into FL to improve it. For instance, the involvement of blockchain in FL improves the underlying communication, such as robustness and availability [34, 35, 36].
- Integrating FL increases the complexity, which scales with the needs of the various mechanisms such as parallelism, data distribution, client selection,

varying training engine, model analysis, data management, bandwidth and resource allocation.

- The currently available frameworks are limited in terms of standardization and customizability, interfering with the search for new solutions to the said challenges.

Therefore, essential details should be considered when working in a complex environment. A framework with a robust structure can solve these problems, but it must have the right level of abstraction; otherwise, it will not be advantageous. To this end, we propose ModularFed, a research-heavy federated learning framework in which we aim to provide a complete set of extendable tools capable of easing the integration of applications in the FL domain, reducing the implementation times while withholding future projects' extendability and scalability.

In summary, our main contributions are the following:

- Enable individualism, loosen components' dependency and maintain a high degree of flexibility by introducing a protocol-based modular FL architecture. Our framework supports reusability and component independency, which eases the integration of component-specific FL approaches and facilitates work extendability.

- Enhance the framework's adaptability by introducing an intuitive subscription-based architecture for seamless third-parties integration.

- Integrate a modular data control center for clients' data simulation capable of sustaining external/context-specific datasets while providing extendable data distributors.

- Put forward the framework capabilities by conducting extensive experiments within various contexts and configurations. Moreover, we compare the architecture's performance to well-known frameworks such as FedML [37] and TensorFlow Federated (TFF) [38] while taking the latter as a baseline.

Following the proposed architectures and addressing the aforementioned issues, we provide an exhaustive framework to orchestrate FL approaches and authorize interchanging components' capabilities. As such, we open the opportunities for collaboration between multiple techniques while additionally easing the integration of new methodologies. Our experimental results demonstrate our framework's capabilities in supporting various contextual FL problems and scenarios, such as data distribution diversity, client performance, and resource limitation, without affecting performance.

2. Related Work

Various frameworks addressing the integration complexity have been proposed. For instance, TensorFlow Federated [38], by Google, the original creator of FL [9], aims to enhance their TensorFlow engine to support distributed learning. The framework supports essential features such as a subscription to a limited set of events to monitor the execution states making thirds party integration possible. Additionally, the framework incorporates the commonly used datasets and the necessary tools for an effortless start. However, the framework only supports TensorFlow models, making it harder for researchers to integrate FL with models built using another training engine. Another engine-specific FL framework exists, such as FedToch [39], which only supports PyTorch models, and PaddleFL [40] supporting the PaddlePaddle engine.

Contrary to the mentioned frameworks, FedML [37] provides an exhaustive framework that started as a fair benchmarking tool for approaches using federated learning. It integrates a complete federated workflow from client selection, aggregation, and validation. Unlike other approaches, where the tests run only locally, their framework supports message-passing interface-based (MPI) settings, which is the key for parallel client simulation allowing executions to happen within or beyond a single host. Nevertheless, enterprise solutions that embrace the concept of FL, such as PySyft [41], Federated AI Technology Enabler (FATE) [42], and IBM Federated Learning [43], exist. For instance, FATE

and PySyft provide the required mechanism from secure computation protocols based on homomorphic encryption and multi-party computation (MPC) for industries to integrate FL in their framework. However, the lack of the necessary benchmarking tools, the complex integration, and the limited extensibility makes it harder for researchers to integrate industrial-based frameworks for their works.

Table 1: Comparison of the supported features between the available frameworks and ours

KubeFATE	TFF[38] Ours	FedTorch[39]	PySyft[41]	FATE [42]	IBM [43]	FedML[37]	PaddleFL[37]	Ray
Local Execution	✓	✓	✓	✓	✓	✓	✓	✓
Distributed Execution	✓	✓	✓	✓	✓	✓	✓	✓
Virtualization/Real Clients	✗	✓	✓	✓	✓	✓	✓	✓
Communication Protocols	AsyncIO	MPI	-	REST	-	MPI	ZeroMQ	Any
Secure	✓	✗	✓	✓	✓	✓	✓	Any
Modular Components	✗	✗	✗	✗	✗	✗	✗	✓
Built-In Monitoring Tools	✓	✗	✗	✗	✗	✗	✗	✓
Third Party Support	✓	✗	✗	✗	✗	✗	✗	✓
Customizable Data Center	✗	✗	✗	✗	✗	✗	✗	✓
Diverse Data Distributor	✗	✗	✗	✗	✗	✗	✗	✓
Client Simulation	✗	✗	✗	✗	✗	✓	✓	✓

Table 1 provides a detailed comparison between our approach and the current federated frameworks. We emphasize the limitation of the current approaches, which we aim to address using ours:

Lack of modular component support: Modularity, in our case, refers to frameworks' supporting individual building blocks with a separate role assigned for each. Together, modular components form a framework that endorses modular interchange. Due to the lack of support for such a concept, the majority tend to build their own environment instead of working within contextual frameworks with a high learning curve while risking the possibility of not supporting their needs. Consequently, the experimental results might not demonstrate a fair comparison, considering the distinct implementation differences and unmanaged statistical environments. Addressing these issues, we pursue a layered architecture in our framework called the federated abstract layer (FAL). With FAL, we design our components based on carefully crafted protocols while delegating the behaviours to the following layers. Accordingly, protocols are blueprints that

define the interaction between the architecture components, allowing researchers to develop flexible and extensible FL approaches. FAL enables FL to support diverse execution mechanisms, such as distributed execution in virtual containers, real devices, or local parallel execution on the same host. Additionally, FAL facilitates extending approaches with supporting features such as security and bandwidth optimization through model compression modules.

Lack of data management protocols: Current frameworks only support a few well-known datasets used in FL. Built-in tools back up these datasets for easier management, such as dynamic allocation from the cloud or data distribution simulators. However, current frameworks tools are limited to well-known datasets, while incorporating new ones compels designing these tools from scratch. In this regard, we integrate data management as part of the framework while outsourcing the required protocols and APIs to allow further expansion depending on the designer's needs.

Third Party Support: By third parties, we refer to any components with a role not related directly to FL workflow. For instance, external monitoring components such as Tensor-Board and Wandb are considered under this category. Additionally, components built to enhance FL architecture are considered a third party, such as model caching, models' parameters analysis, logging, and others. These tools do not affect or are presented in the federated workflow. However, including them increases the framework's complexity. Thus, frameworks either do not support third-party tools or only support a few well-known ones. In our case, we started our framework design with the concept of third parties by incorporating an observable pattern within our components. As such, a list of third parties, called subscribers, can be attached to any federated application and receive live broadcasts about the state of the execution employed, extending it further with various utilities such as monitoring, logging, or analysis.

3. Federated Learning Framework

Figure 1 provides an overview of the proposed framework architecture. As the figure shows, the framework is a bridge that enables federated learning for applications while supporting well-known technologies to complete its objectives. Three key components define our architecture from the rest and make the bridge adaptable to both external layers: FL Abstraction Layer (FAL), FL Subscribers, and Data Control Center.

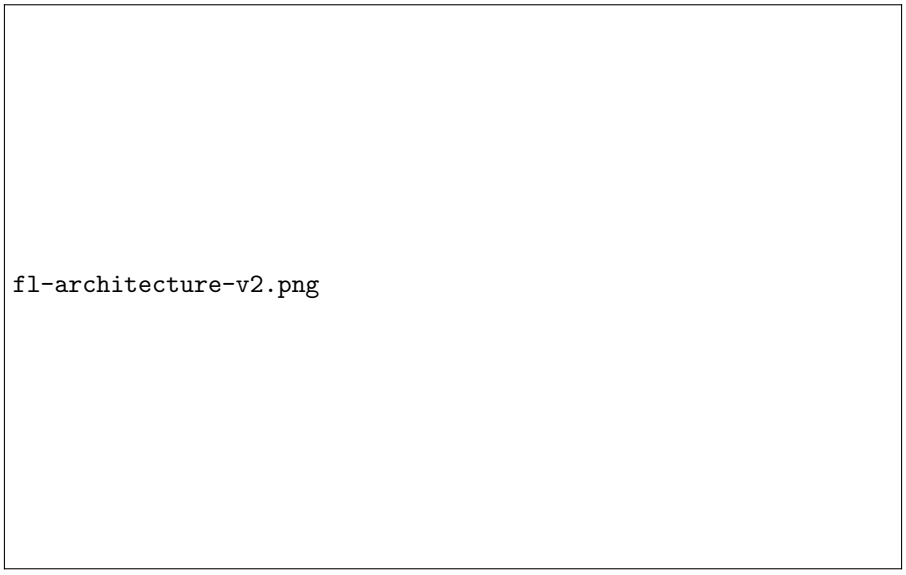


Figure 1: Proposed Layered Architecture

3.1. FL Abstraction Layer

We follow a hierarchical mechanism to qualify extensions and new approaches control over the necessary component dynamically and with minimum modification to the rest of the framework. We standardize FL components through FAL as protocols defining compulsory properties and parameters while delegating the behaviour to the subsequent layers. Federated learning workflow reflects on FAL components and is separated into two higher-order components: Network

Components, comprising the Client and the CL-Manager, to handle the communication and Processing Components, comprising CL-Selector, Aggregator, and Metrics for vital calculations such as aggregation and client selection. Moreover, the core is the glue that manages the FAL and advances with the ordered execution of the FL workflow while taking advantage of the FAL flexibility.

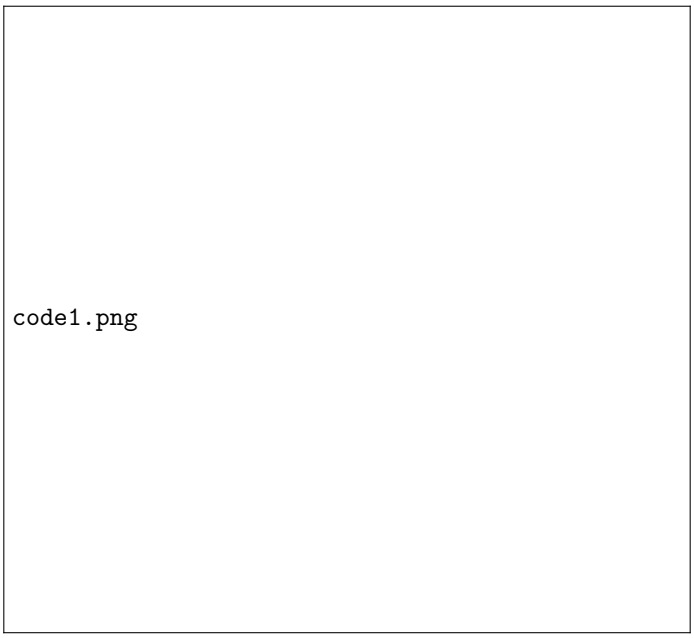


Figure 2: FAL Design Pattern

The Client component is an interface that enables the framework to simulate diverse training contexts allowing it to adapt to the needed scenarios. For instance, a client simulation can be straightforward, such as a PyTorch instance for model training, or more complex, such as a virtual node or a tangible IoT device. We consider the client component as an independent entity from the core module, which opens opportunities for more complex topologies where clients are free from core control while withholding only the minimum specification posed by FAL architecture.

On the other hand, CL-Manager is the middleware between the core and clients, responsible for managing the clients, communicating tasks, and global

model updates. CL-Manager shapes the interaction between the core and the clients authorizing various behaviours such as parallelism, secure integrated network, or further network optimization such as model compression or communication over any available network protocol. Depending on the case, each client works with a distinct Client-Manager. The workers and server can communicate through local threads, MPI, WEB-API, or others. Following a modular architecture, such modules are separately developed following the FAL protocols and given during the creation process. Moreover, choosing the proper CL-Manager relies on the application use case. For instance, when working in an IoT context, a CL-Manager based on WEB-API can handle the high fluctuations of devices' availability. In this case, the server will act similarly to web servers, waiting and resolving remote requests when they arrive. While in contrast, for highly computational and guaranteed environments, for instance, communication between banks or hospitals, MPI can handle fast, efficient and secure communication over ssh for remote data transition to enable parallel execution of locally simulated clients, which might not be supported by the platform ecosystem.

Finally, the core controls the execution of the given interfaces. It accepts any components extending the FAL protocols with additional subscriber components we will discuss in Section 3.2. To keep track of the progress, the FL core creates an FL-Context for each execution. It withholds information regarding the state of the execution, including the global model weights, round number, and round metrics results.

Regardless of what a researcher aims for, an inclusive environment is vital for running tests and obtaining results while comparing them with others. Through FAL, we provide the necessary environment for researchers to overlook the context and concentrate on their targets. Additionally, the modularity of the framework components, which originated from the layered architecture, allows module swapping, minimizes efforts, and secures a fair comparison.

3.2. FL-Subscribers

In this section, we describe FL's standards workflow followed by our adaptable and flexible architecture.

Figure 3 presents the standard approach. The server starts with the initialization, validating the parameters and issuing the first copy of the Global Model. A federated round begins by selecting a set of clients from the available ones. The selected clients receive a copy of the Global Model and proceed with the model training from their local datasets. When the selected clients finish, each sends back the models to the server. The server aggregates the received updates and generates a new, evolved Global Model, substituting the old one. When working in a research environment, part of the dataset is kept to monitor the model evolution and validate the approach's applicability. After each aggregation, the server infers the model metrics from the test dataset and logs them. Eventually, FL stops when reaching the stopping criteria, such as completing a specified number of rounds or achieving a predefined accuracy or loss.

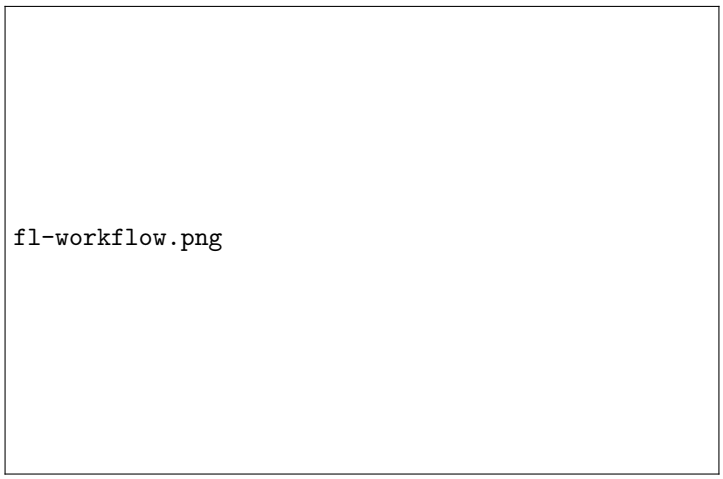


Figure 3: Federated Learning Workflow

However, a typical federated learning application entails different middleware mechanisms such as logging, caching, monitoring the model evolution, drawing charts, saving results, and others. The diversity of these tools is not limited, as

each project needs its own. For example, integrating model evolution to monitor the weight divergence between the clients' parameters will eventually increase the workflow intricacy and complicate its integration with other solutions. Thus, we consider our subscription architecture to solve the mentioned challenges. We built our kernel following the observable software design pattern to achieve our objective. During the initialization step, the FL core registers a list of components subject to the FL Subscriber protocols. During the execution, each subscriber receives, in real-time, broadcasts from the core in the form of updates comprising the states of the intended events. These events have predetermined a priory, and each contains an update bound to specific events in the workflow. Figure 4 contains a detailed representation of the FL workflow core following the integration of the observable pattern. The events cover every step in the workflow; each provides the execution state and distinct information related to the event phase. For example, $bct : trainers\_selected$ provides information about the selected trainer ids, while $bct : round\_finished$ includes the measured metrics of the Global Model after merging the latest updates.

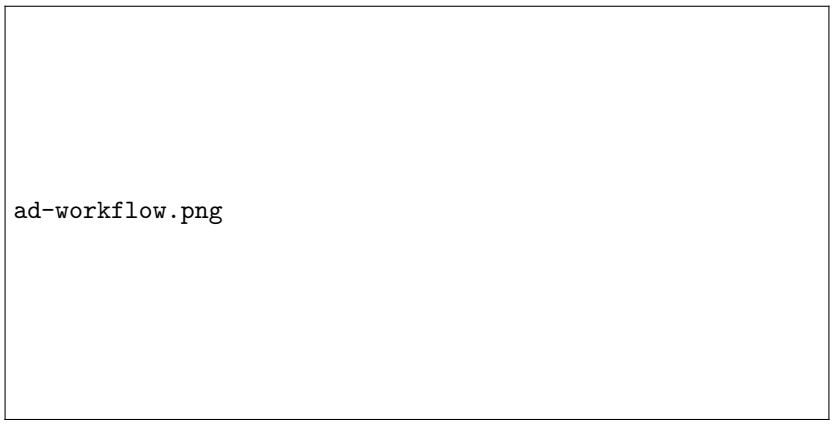


Figure 4: Federated Learning Workflow Including Subscribers

Figure 5 depicts the subscribers' integration into a training procedure. The framework provides a list of subscribers, such as Logging Subscribers, to keep track of the execution progress and estimate the execution time. Caching Sub-

scriber is capable of saving execution checkpoints, allowing the runtime to be resumable in case it shuts down due to unplanned circumstances. Metrics Logger stores the metrics result to SQL database, Markup Based Files or external third-party tools such as TensorBoard or Wandb [44]. Finally, we include analysis tools as subscribers to monitor the model weights evolution or client selection shift after each federated round. Limitless functionalities can be plugged into our framework without altering the core. It is an excellent choice for researchers aiming for collaboration, extending their work or carrying out their methodologies to other approaches.

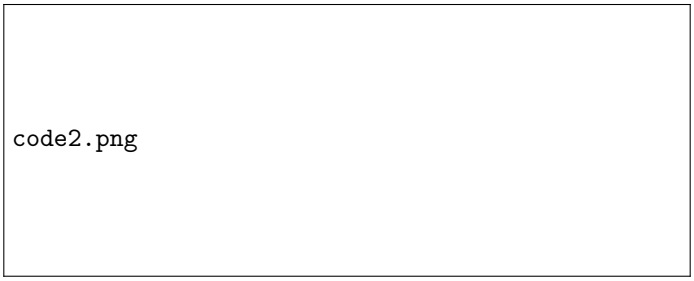


Figure 5: Subscription Design Pattern

3.3. Data-Center

Providing solutions for federated learning requires working with different datasets collected from various sources. Additionally, most experiments involve working with diverse clients' data distributions between to check if the solution is tolerant to the Non-IID problem. Simulating various Non-IID client behaviour is challenging when combined with the diversity of the datasets. Thus, we aim through our Data-Center to provide the necessary protocols addressing the aforementioned problems. Figure 1 introduces our Data-Center, as an underlying methodology supporting our framework with its three key components: Containers, Distributors, and Providers.

Integrating data **Containers** unifies the data objects under one protocol across the framework, alleviating the complexity of working with multiple sources.

Moreover, the core integrates the containers in its workflow, allowing it to handle some dataset routines such as train/test split, batching and type conversion. **Providers** represent the raw dataset sources used for the client simulation. The source can be anything from cloud storage, databases, data streams, local files, real devices, and others. The protocols imply outsourcing the raw data to the framework as Containers, which can be used directly to simulate the client's behaviours. Finally, it is essential to demonstrate the capability of any approach against the statistical difficulties caused by the federated learning architecture. With the **Distributors** component, we aim to standardize the distribution strategies under one protocol capable of simulating most of the Non-IID/IID scenarios. The behaviour of the distributor varies depending on the implementation. In the following, we explain in detail the currently supported distribution while enclosing heatmaps images which present an example of the data distributed to each client following the usage of a distributor. In the images, the Y-axis represents the labels, the X-axis represents the clients, and the intersection between X and Y represents the number of records a label y a client x container withholds. We built a dataset of records identified by 10 labels distributed to 20 clients for these representations.

ShardDistributor: Introduced by google [9], such distribution allows experimenting under strict Non-IId environments. The data is split into shards S of equal size. Each shard contains a fixed number of raw records with the same label. Afterward, the shards are distributed to clients, each receiving a predetermined number of shards selected randomly. In [9] use cases, each client receives two shards of 300 records each, indicating that each client has at most two labels in its local datasets, creating a highly Non-IID scenario. It is possible to efficiently control the IIDness severity in this case as it depends on a single value: how many shards S each client receives. The shard distribution is represented in Figure 6c. Each shard S is fixed with 300 records, and 2 of these shards are distributed to 20 clients.

LabelDistributor: Similar to the shard distribution but with more control over the number of labels L distributed to each client. Unlike Shard Distribu-

(a) Dirichlet Distribution with $\alpha = 0.5$

(b) Dirichlet Distribution with $\alpha = 10$

(c) Shard Distribution with $S = 2$

(d) Label Distribution with $L = 1$

(e) Label Distribution with $L = 10$

(f) Unique Distribution

Figure 6: MNIST FL Experiments on Label, Dirichlet, Shard, and Unique Distributions

tion, each client is guaranteed to have the specified number of labels in their datasets. Additionally, it's possible to simulate client distribution with different label sizes, which is irrelevant when using Shard Distribution. In Figure 6e and 6d, we show the results of a label distribution to 20 clients, each receiving 600 records. While Figure 6e represents an IID distribution where all the clients where all clients receive an equal amount of data for each label, in Figure 6d, we show a highly Non-IID distribution where each client hold records from only one label.

UniqueDistributor: Using a unique distributor, it is possible to create a particular type of severe Non-IID case in which each client have a dataset with records from a single, unique label/class. Unlike the rest of the distributors,

where the same record label/class might exist on multiple client datasets, the Unique distributor guarantees that the single label in one client dataset does not exist in another. The Unique distribution is considered a serious difficulty of Non-IIDness and occurs in studies focusing on clients' penalization, such as behavioural analysis [9]. Such a distribution is presented in Figure 6f. Different from the highly Non-IID case presented in 6d where each client holds records from only one label, in the Unique case, two clients holding the same label does not exists. As a result, it is only possible to show the data distribution to 10 clients since our dataset has only ten labels.

DirichletDistributor: applied by [37], used to generate a vector of samples where each signifies the percentage of the representative label index of the total data size, which can be assigned randomly during the distribution. In this case, it is possible to create a form of data unbalancement in terms of local labels across different clients. For instance, we can use Dirichlet to create a client that possesses 10% of its total records labelled 0, and the rest labelled 1, while another client possesses 85% of records labelled 0 and the rest labelled 1. Numbers vectors: [0.1,0.9] and [0.85,0.15] are two samples drawn from the Dirichlet Distribution with predefined alpha (α) values that denote the data skewness. Such distribution can represent real-life scenarios where data differ not only in terms of general datasets size between clients but also in terms of each label size. In Figure 6b we show an IID Dirichlet Distribution with $\alpha = 10$ where each client holds almost every label in equal size with little deviation. In Figure 6a, we show a Non-IID case where $\alpha = 0.5$. In this case, a significant number of clients with missing labels combined with a huge divergence in label size across multiple clients.

The combination of the aforementioned mechanics delivers a smooth and straightforward interaction between the datasets and the framework, qualifying the researchers' complete and direct control over the data. For example, unlike other frameworks, the datasets are independent of the core, allowing researchers to take advantage of these benefits when supplementing their datasets. Moreover, the framework supports personalized distributors and providers as long as

they integrate the corresponding protocols.

4. Datasets, Benchmarks & Experiments

4.1. Datasets

To test our framework and confirm its validity, we ran multiple tests, including working with various datasets, models, and configurations. We examine the framework performance on various datasets, including MNIST, FEMNIST, and CIFAR10, covering use cases that could exist in any classification problem. MNIST is a digit dataset which consists of written digit images of numbers from 0 to 9 in two channels. The dataset contains 70k records of 24*24 pixels. On the other hand, FEMNIST is another similar dataset to MNIST but with increased difficulty and more labels, including 28*28 pixel images of letters and digits, forming a dataset with 62 labels distributed between 671k records. The main difference between MNIST and FEMNIST is in the active pixel coverage in both dataset images. Figure 7 shows a comparison between the same digit image. Having fewer active pixels in FEMNIST makes that dataset more challenging and requires a more capable model, which requires additional time and resources. CIFAR10 is a categorical classification dataset containing images of animals and objects in three channels. The main objective is to create a model capable of differentiating between them. The dataset contains 60k 32*32*32 images distributed into 10 labels.

Regarding the global model configuration, we used Logistic-Regression for MNIST, a simple and fast model with fewer parameters than others with convoluted configurations. As a result, low-end device trains model faster due to the reduced allocated resources and less bandwidth consumption at the cost of not reaching higher accuracy. It is always the decision of precision vs performance in federated learning where we sacrifice precision for a faster and lightweight model or the contrary. CNN is used for FEMNIST and CIFAR10, each with a different configuration adapting to the differences in the image size.

Figure 7: Comparison between MNIST and FEMNIST images.

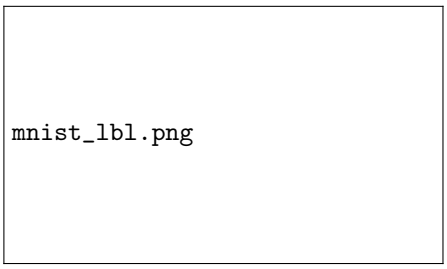
4.2. Configuration Parameters

Regarding the framework configuration, it consists of the following parameters:

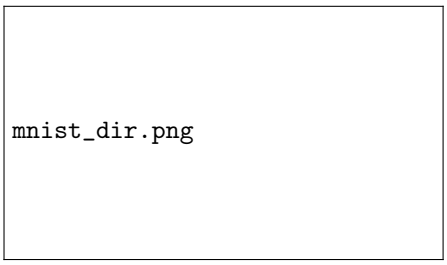
- Data Distributor: It defines how the data is distributed to the simulated clients. This parameter dramatically impacts the model accuracy, such as if the dataset is distributed in an IID or Non-IID manner. The latter's impact is determined by its severity which is discussed in the previous sections.

- CL-Manager: Affect the framework speed and portray the communication between the FL server with its clients. There are two CL-Manager used in the experiments, SequentialManager, which simulates synchronous trainers running in sequence one after another, and an MPIManager, which simulates parallel trainers running concurrently.

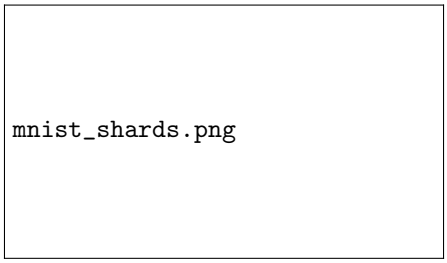
- Epoch: Have a significant impact on the framework accuracy. An important term is introduced by [9], which calls a single epoch and no-batching federated learning FedSGD in which the trainers train their model running over all the data only once and send the model back to the server. In our experiments, we tested the model on both 1 Epoch (FedSGD) and



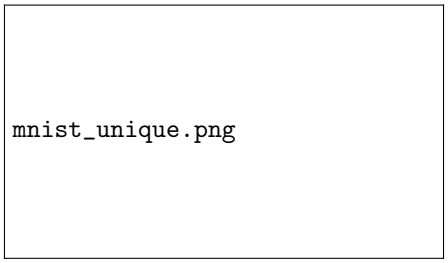
(a) Label Distribution on MNIST Dataset



(b) Dirichlet Distribution on MNIST Dataset



(c) Shard Distribution on MNIST Dataset



(d) Unique Distribution on MNIST Dataset

Figure 8: MNIST FL Experiments on Label, Dirichlet, Shard, and Unique Distributions

50 Epoch to show the impact of the parameters on the model accuracy evolution.

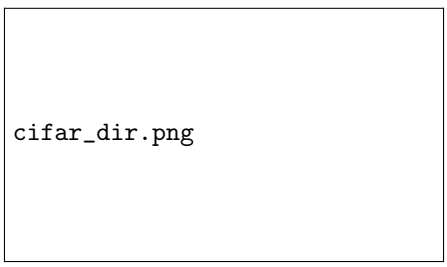
- Client Selection: Throughout our experiment, we used a simple client selection procedure: selecting a predefined random number of clients for each round. This parameter simulates real-case scenarios where most clients are not simultaneously available.

- Learning Rate: A hyper-parameter set for model training to control the rate of the model's parameters update and how much the model accepts from the new weights through a value ranging between 0^+ and 1. Choosing the optimal learning rate can be achieved through small tests (a few rounds) checking the model evolution under various learning rate values. We did these tests beforehand and picked the optimal results.

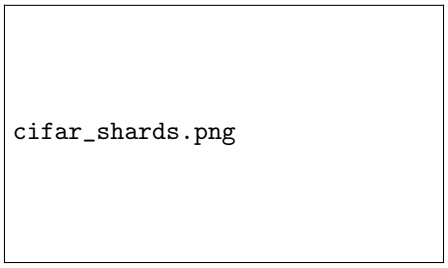
4.3. Benchmarks & Experimental Results

Our primary objective is to provide a platform capable of incorporating the huge configurations varieties in the FL environment. Additionally, we aim to effortlessly replicate the main issues in the FL context allowing researchers to solve them in a straightforward and organized manner. In the following experiments, we test and validate our FL framework under various data distributions, which are the roots of the statistical issues in FL, and client-server approaches.

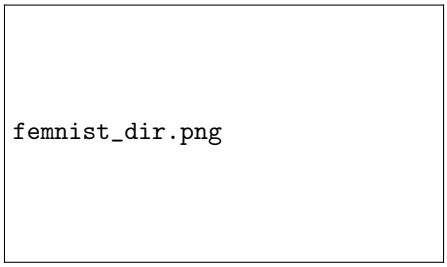
Figures in 8 and 9 present the evolution of the global model accuracy when experimenting with the impact of both data distribution and epochs. Every test includes FedSGD (single epoch/round) and FedAVG (multiple epochs/round) to show the epoch's influence. Regarding the distribution, we experiment on MNIST using every mentioned distribution to demonstrate and highlight its influence on accuracy. Respecting the limited article length, for the subsequent datasets, we show only Dirichlet due to its capabilities of simulating real-life scenarios and Shard Distribution which is mainly used in FL works. Furthermore, each experiment includes both IID and Non-IID distributions. For Label Distribution, $L = 10$ is considered IID, where each client receives data of equal size from every Label, while the $L = 1$ is considered the Non-IID case, where each client receives only one Label. For Dirichlet Distribution, we act on the α value, which controls the data skewness of the randomly generated float values. A higher α value results in a notable data disparity between clients, which reduces the data Non-IIDness. Thus, we chose $\alpha = 10$ as an IID representative and $\alpha = 0.5$ to portray a Non-IID distribution. Finally, for Shard Distribution experiments, we consider $S = 2$ as Non-IID in which each client has at most two labels of 300 records each and $S = 5$ as an IID case where each client has at least five labels of 300 records each. The total number of clients is different depending on the subsequent distribution. For instance, working with Label and Dirichlet Distributions, we can predefine a fixed number of clients, of which we chose 100. Shard distribution depends on the number of records and the shard size. For instance, MNIST, with 60k records divided into 300 records/shards, results in 200 shards. Thus, we have 100 clients when $S = 2$ compared to 40



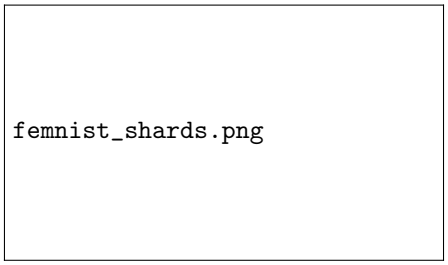
(a) Dirichlet Distribution on CIFAR10 Dataset



(b) Shards Distribution on CIFAR10 Dataset



(c) Dirichlet Distribution on FEMNIST Dataset



(d) Shards Distribution on FEMNIST Dataset

Figure 9: CIFAR and FEMNIST FL Experiments on Dirichlet and Shard Distributions

when $S = 5$. As for Unique Distribution, we are limited by the total number of Labels. For example, in MNIST, since the dataset contains only 10 Labels, we can have at most ten clients. For our experiments, we employed a random client selector in which we selected $CR = 10$ clients for each round.

Figure 8a shows the results of MNIST Label Distribution when using Logistic-Regression as the base model configuration. When working with the MNIST dataset, the impact of the distribution is negligible. It can be solved with additional rounds because the dataset is structured with simple and properly filtered images. However, there are still some noticeable differences. With $E = 50 : L = 1$, we notice huge spikes in the accuracies, which indicate that the model is struggling to find the optimal parameters. The intensity of spikes decreased with the subsequent rounds but did not converge even after 1000 rounds. Such a scenario led to additional computational and resource consumption compared to $E = 50 : L = 10$, which converged at 90% after 200 rounds. Nevertheless, the impact of epochs is more noticeable in the next experiment.

For $E = 1$, the accuracy was lower than $E = 50$. Figure 8b shows the results under Dirichlet Distribution while Figure 8c show them under Shard Distribution. it is hard to notice major differences as both follow the same pattern. All of the mentioned results show that a high epoch rate follows an increase in the model performance. However, a higher epoch does not strictly reflect better accuracy in every case. In Figure 8d, we show a severe case of Non-IIDness in which each client has a unique single label in their dataset. In this special case, FedSGD performed better than FedAVG as well as less spike, which indicates a stable improvement. However, this result is only achieved when using LogisticRegression as a model configuration on the MNIST dataset, which might vary when using other configurations.

In the next batch, we present our experiments using FEMNIST and CIFAR10 datasets. Unlike MNIST, the datasets under consideration contain considerably more information regarding the number of pixels and their variety. For FEMNIST, the image is smaller, while in CIFAR10, the images have three channels instead of 2 in FEMNIST and MNIST. Figure 9a and 9b show the influence of Dirchilet and Shard distribution on the accuracy of the federated learning global model using the CIFAR10 dataset. The number of Epochs has a significant impact on these experiments. Under FedSGD, the experiments under both distributions started with lower accuracy and reached a higher accuracy than the $E = 50$ experiments. Additionally, both experiments' accuracies suffered from the Non-IID distribution. Regarding FEMNSIT dataset experiments, Figure 9d shows the data distribution of FEMNIST when using the Shard Distributor, while Figure 9c portrays the Dirichlet distributor. Under the FedSGD environment, FEMNIST suffers from an apparent problem: the accuracy did not improve in either the IID or Non-IID context. However, increasing the number of Epochs sustains the accuracy improvements. Under Dirichlet influence, the Non-IID distribution started with a worse accuracy while reaching similar results at the end of the experiments. The difference between IID and Non-IID is evident in the shard distribution, Figure 9d.

We compile all of our experimental results in Table 2 showing the final

accuracy and loss at the end of each experiment when using $E = 1$ and $E = 50$ under both IID and Non-IID distributions.

Table 2: Last Accuracy and Loss of federated learning experiments when using $E = 1$ and $E = 50$ under IID and Non-IID experiments

			Results			
			ACC		**LOSS**	
			E=1	**E=50**	**E=1**	**E=50**
MNIST	Label	IID	0.8415	0.899	1.621	1.556
		N-IID	0.854	0.867	1.62	1.637
	Shard	IID	0.859	0.9002	1.593	1.555
		N-IID	0.852	0.911	1.6214	1.573
	Dirichlet	IID	0.849	0.903	1.612	1.554
		N-IID	0.845	0.906	1.613	1.56
	Unique	N-IID	0.896	0.784	1.582	1.727
CIFAR	Shard	IID	0.519	0.438	1.419	5.104
		N-IID	0.323	0.394	1.996	3.175
	Dirichlet	IID	0.4946	0.456	1.4	5.769
		N-IID	0.433	0.415	1.584	4.719
FEMNIST	Shard	IID	0.106	0.585	4.068	3.579
		N-IID	0.0509	0.498	4.094	3.691
	Dirichlet	IID	0.326	0.889	3.858	3.265
		N-IID	0.203	0.889	3.951	3.26

Different from our previous experiments, In Figure 10 we highlight the bandwidth cost of using different model configurations showing the impact on the model accuracy. The Figure shows the evolution of the accuracy throughout the federated learning rounds, and the accumulative bandwidth cost annotated for every 100 rounds. The model configurations in use are CNN 2 Layers and LogisticRegression. For the hyper-parameters, we used FedSGD on the MNIST dataset distributed to 100 clients using Dirichlet Distributor with $\alpha = 10$. Ten clients are randomly selected to train a model using the provided configuration in each round. Since we are not using any compression algorithm, the models' weight size is constant. In this experiment, using LogisticRegression, we achieved an accuracy of 88% after 1000 rounds which cost a total of 4.236 MB, compared to 90% accuracy reached when using CNN achieved after ten rounds with a total cost of 0.08 MB. In this case, an exemplary model configuration can reduce the total bandwidth cost, although it costs more per client per round.

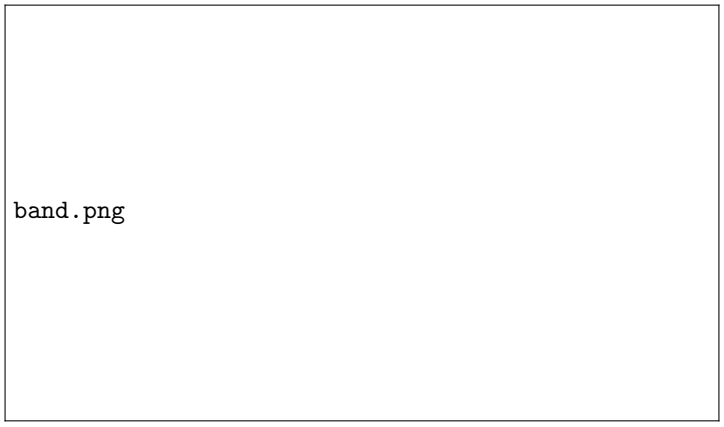


Figure 10: Bandwidth transmission cost comparison between CNN and LogisticRegression (LogR) model configuration

In Figure 11, we show the impact of the client selector on the global model accuracy after each round. For this experiment, we used Label Distributor with $L = 2$ to create a Non-IID data distribution between clients coupled with FedSGD configuration and a client ratio of 10. We compare between random client selector with a cluster-based client selector. In the latter, we start with

an initialization round, asking each client to train a model and send it to the server. The selector will use the model as input to identify the client cluster using the K-Means algorithm with a parameter K, which refers to the number of clusters. During each round, the clustering algorithm selects a predefined number of clients from each cluster to participate. For instance, for a client ratio of ten, and $K = 5$, we select two clients from each cluster. Using such an approach makes it possible to have a variety of model weights for each round, which can be used to solve the Non-IID distribution issues. Overall, the results in Figure 11 demonstrate the capabilities of the cluster selector to achieve better and more stable results than the random selector.

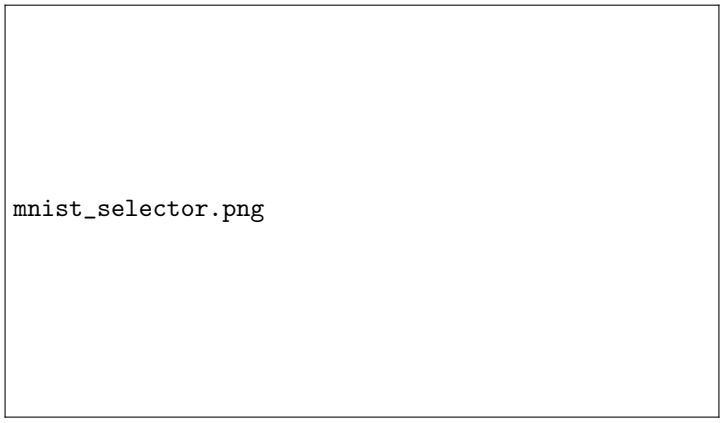
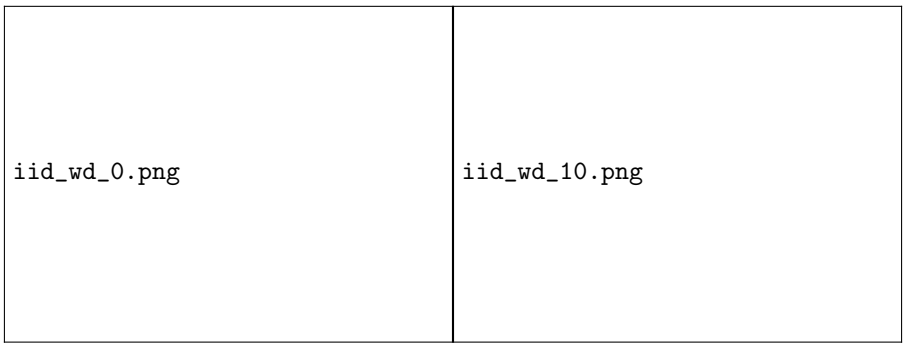


Figure 11: Comparison between Cluster Selector and Random Selector using MNIST dataset

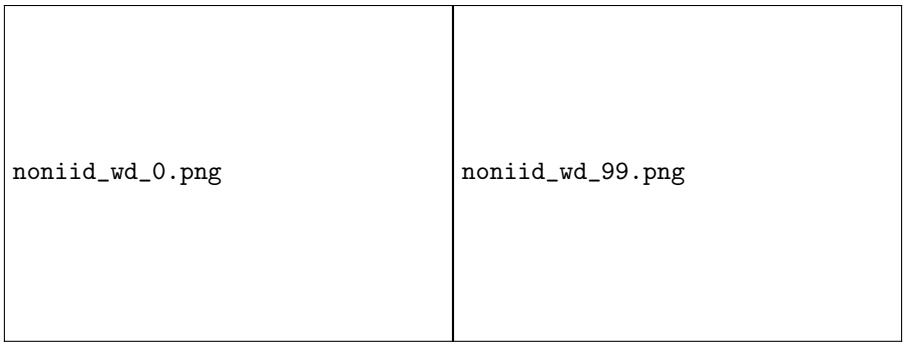
4.4. Weight Divergence Analysis Module

Figures 12 depict integrating a complex weight analysis mechanism into any federated learning approach. In this experiment, the module monitors the evolution of weight divergence between clients' weights in both IID in Figures 12a, 12b and Non-IID in Figures 12c, 12d. In each figure, each plot line represents the flattened weights of an MNIST client. Due to the size of the weights, principal component analysis (PCA) is used to reduce the number of representable values.



(a) IID
Label Distribution $R = 0$

(b) IID
Label Distribution $R = 10$



(c) Non-IID
Label Distribution $R = 0$

(d) Non-IID
Label Distribution $R = 99$

Figure 12: Models' Weights Representation of 100 IID/Non-IID Clients Using Weight Divergence Analysis Module

Comparing the IID experiment 12a to the Non-IID 12c at round 0, we can notice a difference in the weight divergence between the clients. When following an IID distribution between clients' data, there tend to exist similarities between the generated models' weights. The appearing weight divergences subside to almost identical weights after 10 rounds, as shown in Figure 12b.

On the other hand, the Non-IID clients in Figure 12c show a higher weight divergence between them. The weight divergence is still evident even after 99 rounds, as shown in Figure 12d compared to the IID case. This weight representation can imply a direct connection between the federated learning performance and the weight divergence that exists due to the data distribution.

The higher the weight divergence, the more it has adverse effects on the global model in terms of accuracy and convergence rate.

4.5. Performance Analysis

In this section, we compare our approach to TensorFlow Federated (TFF) [38], and FedML [37]. TFF contains minimal configuration with limited utilities, while the rest provides various tools, including client selection, parallelization through various communication protocols, virtualization, data distributors and others. The experiments are executed on a host using a graphical processing unit under the Linux operating system. The host configuration contains 16GB RAM and 6GB VRAM capable of withholding both datasets and the model configuration. We employed Logistic Regression as a model configuration for this experiment since we are only comparing the performance of the execution between each of the frameworks. The MNIST dataset is used for the training procedure, distributed to 100 clients using the same distribution method for each. Automatic download and dataset extraction are supported by the listed frameworks. Moreover, we select 4 random clients in each round to proceed with the local training. We ensured the parallel execution of clients using MPI with 5 parallel instances (1 Server, 4 Workers) in FedML and Ours, while TFF uses its built-in threading enabler "nest_asyncio." After each experiment, we reset the operating system to ensure that each framework environment does not affect the other experiments. Our emphasis is on the execution speed and the bootstrap time. The execution speed examines the efficient utilization of the host resources, while the bootstrap time investigates the framework's efficiency in handling the data distribution and loading data into memory. In the following experiments, all frameworks achieve similar accuracy and loss.

Figure 13 compares the execution time in seconds throughout the rounds. As we notice in the figure, our framework provides a robust architecture with numerous capabilities at an insignificant performance cost. Our approach finished processing 500 rounds in 180 seconds, taking an average of 0.36 seconds/round compared to TFF, taking 100 seconds, averaging 0.2 seconds/round. In con-

trast, due to poor management of resources and multiple loading of datasets into memory on each instance, FedML execution takes significantly more time to finish the execution, taking a total of 2875 seconds while averaging 5.75 seconds/round.

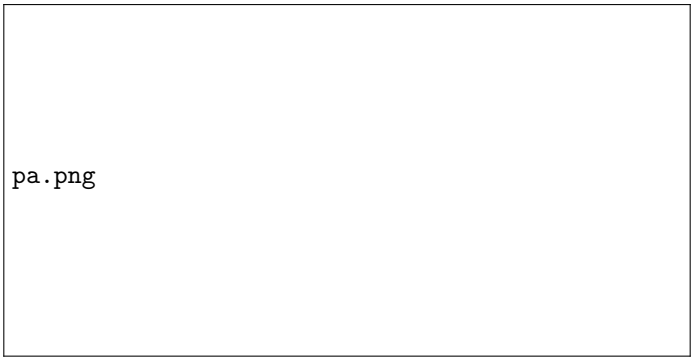
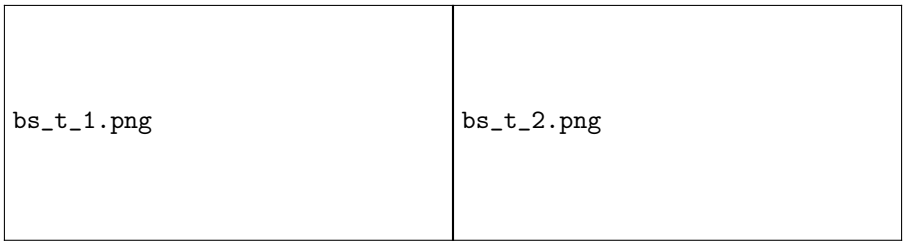


Figure 13: Performance Analysis Comparison Between Our approach, TFF and FedML

Furthermore, Bootstrap Time, as shown in Figure 14, is also logged during the beginning of the execution, measuring the time needed for each framework to load all its components and dataset to memory and distribute the dataset to clients. TFF uses SQLite to manage the data distribution between clients. Our approach includes a data control center which handles the clients' data distributions. In contrast, for the server and for each new worker, FedML loads the full dataset into memory, and clients take their share of the datasets when they start training. Such a practice takes a toll on the memory if it's running under one host. It was made to support only multi-device configuration, which limits the benefit of MPI, enabling fast and parallel computation on a single machine. Our experiments include the first execution time, available in Figure 14a, and the subsequent execution time in Figure 14b. Thus, we can emphasize the improvement we made in our approach. Rather than distributing the data each time we proceed with a new experiment, the results of the data distribution following any previous transformation are cached into a file that can be loaded directly when we intend to work with it again. Such an approach significantly reduces the bootstrap time, which is more crucial when working on

bigger datasets on hosts with few available resources. TFF, on the other hand, does not provide any additional distribution, or distributor tools, besides the default one. On the first execution, our approach took 9 seconds, reduced to 3.98 in the second execution compared to 5.44 for TFF and 242.4 for FedML.



(a) Bootstrap Time In The First Execution (In seconds)

(b) Boostrap Time In The Subsequent Executions (In seconds)

Figure 14: Bootstrap Time Comparison Between Our approach, TFF and FedML

5. Conclusion

With federated learning getting acknowledged for its premise, there is a lack of fundamental conventions in available frameworks. We aim to solve this issue by introducing our protocol-based layered architecture, with modular support allowing our framework to act as a complete ecosystem. Using our modular implementation, we consider FL functionalities as independent modules enabling components' individuality during the development and pushing for a collaborative environment and modules' reusability. Following such an architecture and an observable pattern structure, our framework can adapt to any situation's requirements while eliminating most limitations. Moreover, our framework is built from scratch to support projects' extendability while providing the necessary tools to replicate most FL-related issues. We laid the groundwork through ModularFed, and we plan to further enhance it in the future by integrating various supporting technologies as modules. Technologies such as traceability through blockchain or components deployments and orchestration through Kubernetes microservices. Finally, we experimented with the framework's validity and flexibility under various FL scenarios, including distribution issues, network consumption, and client selection through quality assessment. Our performance analysis emphasizes the benefits we can get out of the architecture at a small performance cost when compared to a basic approach. Our framework will be constantly maintained by introducing new features and various new modules following the latest directions.

References

[1] T. Nishio, R. Yonetani, Client selection for federated learning with heterogeneous resources in mobile edge, in: ICC 2019-2019 IEEE international conference on communications (ICC), IEEE, 2019, pp. 1–7.

[2] Y. Zhan, P. Li, Z. Qu, D. Zeng, S. Guo, A learning-based incentive mechanism for federated learning, IEEE Internet of Things Journal 7 (7) (2020) 6360–6368. doi:10.1109/JIOT.2020.2967772.

[3] S. Abdulrahman, H. Tout, A. Mourad, C. Talhi, Fedmccs: Multicriteria client selection model for optimal iot federated learning, IEEE Internet of Things Journal 8 (6) (2021) 4723–4735. doi:10.1109/JIOT.2020.3028742.

[4] W. Zhang, X. Wang, P. Zhou, W. Wu, X. Zhang, Client selection for federated learning with non-iid data in mobile edge computing, IEEE Access 9 (2021) 24462–24474. doi:10.1109/ACCESS.2021.3056919.

[5] S. A. Rahman, H. Tout, C. Talhi, A. Mourad, Internet of things intrusion detection: Centralized, on-device, or federated learning?, IEEE Network 34 (6) (2020) 310–317. doi:10.1109/MNET.011.2000286.

[6] O. A. Wahab, A. Mourad, H. Otrok, T. Taleb, Federated machine learning: Survey, multi-level classification, desirable criteria and future directions in communication and networking systems, IEEE Communications Surveys & Tutorials 23 (2) (2021) 1342–1397. doi:10.1109/COMST.2021.3058573.

[7] S. Abdulrahman, H. Tout, H. Ould-Slimane, A. Mourad, C. Talhi, M. Guizani, A survey on federated learning: The journey from centralized to distributed on-site learning and beyond, IEEE Internet of Things Journal 8 (7) (2021) 5476–5497. doi:10.1109/JIOT.2020.3030072.

[8] S. Arisdakessian, O. A. Wahab, A. Mourad, H. Otrok, M. Guizani, A survey on iot intrusion detection: Federated learning, game theory, social psychology and explainable ai as future directions, IEEE Internet of Things Journal (2022) 1–1doi:10.1109/JIOT.2022.3203249.

[9] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, B. A. y. Arcas, Communication-efficient learning of deep networks from decentralized data (2016). doi:10.48550/ARXIV.1602.05629.
URL https://arxiv.org/abs/1602.05629

[10] C. Briggs, Z. Fan, P. Andras, Federated learning with hierarchical clustering of local updates to improve training on non-iid data, in: 2020 International

Joint Conference on Neural Networks (IJCNN), 2020, pp. 1–9. doi:10.1109/IJCNN48605.2020.9207469.

[11] F. Sattler, K.-R. Müller, W. Samek, Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints, IEEE Transactions on Neural Networks and Learning Systems 32 (8) (2021) 3710–3722. doi:10.1109/TNNLS.2020.3015958.

[12] Y. Kim, E. A. Hakim, J. Haraldson, H. Eriksson, J. M. B. da Silva, C. Fischione, Dynamic clustering in federated learning, in: ICC 2021 - IEEE International Conference on Communications, 2021, pp. 1–6. doi:10.1109/ICC42927.2021.9500877.

[13] C. Li, G. Li, P. K. Varshney, Federated learning with soft clustering, IEEE Internet of Things Journal 9 (10) (2022) 7773–7782. doi:10.1109/JIOT.2021.3113927.

[14] H. Wang, M. Yurochkin, Y. Sun, D. S. Papailiopoulos, Y. Khazaeni, Federated learning with matched averaging, CoRR abs/2002.06440 (2020). arXiv:2002.06440.
URL https://arxiv.org/abs/2002.06440

[15] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, A. T. Suresh, SCAFFOLD: stochastic controlled averaging for on-device federated learning, CoRR abs/1910.06378 (2019). arXiv:1910.06378.
URL http://arxiv.org/abs/1910.06378

[16] A. Reisizadeh, F. Farnia, R. Pedarsani, A. Jadbabaie, Robust federated learning: The case of affine distribution shifts, CoRR abs/2006.08907 (2020). arXiv:2006.08907.
URL https://arxiv.org/abs/2006.08907

[17] M. Wazzeh, H. Ould-Slimane, C. Talhi, A. Mourad, M. Guizani, Privacy-preserving continuous authentication for mobile and iot systems using

warmup-based federated learning, IEEE Network (2022) 1–7doi:10.1109/MNET.121.2200099.

[18] G. Sun, Y. Cong, J. Dong, Q. Wang, L. Lyu, J. Liu, Data poisoning attacks on federated machine learning, IEEE Internet of Things Journal 9 (13) (2022) 11365–11375. doi:10.1109/JIOT.2021.3128646.

[19] V. Tolpegin, S. Truex, M. E. Gursoy, L. Liu, Data poisoning attacks against federated learning systems, in: L. Chen, N. Li, K. Liang, S. Schneider (Eds.), Computer Security – ESORICS 2020, Springer International Publishing, Cham, 2020, pp. 480–501.

[20] Z. Chen, P. Tian, W. Liao, W. Yu, Zero knowledge clustering based adversarial mitigation in heterogeneous federated learning, IEEE Transactions on Network Science and Engineering 8 (2) (2021) 1070–1083. doi:10.1109/TNSE.2020.3002796.

[21] S. K. Lo, Y. Liu, Q. Lu, C. Wang, X. Xu, H.-Y. Paik, L. Zhu, Towards trustworthy ai: Blockchain-based architecture design for accountability and fairness of federated learning systems, IEEE Internet of Things Journal (2022) 1–1doi:10.1109/JIOT.2022.3144450.

[22] S. Otoum, I. Al Ridhawi, H. T. Mouftah, Blockchain-supported federated learning for trustworthy vehicular networks, in: GLOBECOM 2020 - 2020 IEEE Global Communications Conference, 2020, pp. 1–6. doi:10.1109/GLOBECOM42002.2020.9322159.

[23] M. H. ur Rehman, K. Salah, E. Damiani, D. Svetinovic, Towards blockchain-based reputation-aware federated learning, in: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications Workshops (INFOCOM WKSHPS), 2020, pp. 183–188. doi:10.1109/INFOCOMWKSHPS50562.2020.9163027.

[24] Label inference attacks against vertical federated learning, in: 31st USENIX Security Symposium (USENIX Security 22), USENIX Associa-

tion, Boston, MA, 2022.
URL https://www.usenix.org/conference/usenixsecurity22/presentation/fu-chong

[25] Y. Gu, Y. Bai, S. Xu, Cs-mia: Membership inference attack based on prediction confidence series in federated learning, Journal of Information Security and Applications 67 (2022) 103201. doi:https://doi.org/10.1016/j.jisa.2022.103201.
URL https://www.sciencedirect.com/science/article/pii/S2214212622000801

[26] F. Sattler, S. Wiedemann, K.-R. Müller, W. Samek, Robust and communication-efficient federated learning from non-i.i.d. data, IEEE Transactions on Neural Networks and Learning Systems 31 (9) (2020) 3400–3413. doi:10.1109/TNNLS.2019.2944481.

[27] L. WANG, W. WANG, B. LI, Cmfl: Mitigating communication overhead for federated learning, in: 2019 IEEE 39th International Conference on Distributed Computing Systems (ICDCS), 2019, pp. 954–964. doi:10.1109/ICDCS.2019.00099.

[28] L. Liu, J. Zhang, S. Song, K. B. Letaief, Client-edge-cloud hierarchical federated learning, in: ICC 2020 - 2020 IEEE International Conference on Communications (ICC), 2020, pp. 1–6. doi:10.1109/ICC40277.2020.9148862.

[29] B. Luo, X. Li, S. Wang, J. Huang, L. Tassiulas, Cost-effective federated learning design, in: IEEE INFOCOM 2021 - IEEE Conference on Computer Communications, 2021, pp. 1–10. doi:10.1109/INFOCOM42981.2021.9488679.

[30] Z. Yang, M. Chen, W. Saad, C. S. Hong, M. Shikh-Bahaei, Energy efficient federated learning over wireless communication networks, IEEE Transactions on Wireless Communications 20 (3) (2021) 1935–1949. doi:10.1109/TWC.2020.3037554.

[31] Q. Zeng, Y. Du, K. Huang, K. K. Leung, Energy-efficient resource management for federated edge learning with CPU-GPU heterogeneous computing, CoRR abs/2007.07122 (2020). arXiv:2007.07122.
URL https://arxiv.org/abs/2007.07122

[32] H. Wang, Z. Kaplan, D. Niu, B. Li, Optimizing federated learning on non-iid data with reinforcement learning, in: IEEE INFOCOM 2020 - IEEE Conference on Computer Communications, 2020, pp. 1698–1707. doi:10.1109/INFOCOM41043.2020.9155494.

[33] A. Hammoud, H. Otrok, A. Mourad, Z. Dziong, On demand fog federations for horizontal federated learning in iov, IEEE Transactions on Network and Service Management (2022) 1–1doi:10.1109/TNSM.2022.3172370.

[34] D. C. Nguyen, M. Ding, Q.-V. Pham, P. N. Pathirana, L. B. Le, A. Seneviratne, J. Li, D. Niyato, H. V. Poor, Federated learning meets blockchain in edge computing: Opportunities and challenges, IEEE Internet of Things Journal 8 (16) (2021) 12806–12825. doi:10.1109/JIOT.2021.3072611.

[35] H. Kim, J. Park, M. Bennis, S.-L. Kim, Blockchained on-device federated learning, IEEE Communications Letters 24 (6) (2020) 1279–1283. doi:10.1109/LCOMM.2019.2921755.

[36] S. R. Pokhrel, J. Choi, Federated learning with blockchain for autonomous vehicles: Analysis and design challenges, IEEE Transactions on Communications 68 (8) (2020) 4734–4746. doi:10.1109/TCOMM.2020.2990686.

[37] C. He, S. Li, J. So, M. Zhang, H. Wang, X. Wang, P. Vepakomma, A. Singh, H. Qiu, L. Shen, P. Zhao, Y. Kang, Y. Liu, R. Raskar, Q. Yang, M. Annavaram, S. Avestimehr, Fedml: A research library and benchmark for federated machine learning, CoRR abs/2007.13518 (2020). arXiv:2007.13518.
URL https://arxiv.org/abs/2007.13518

[38] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, X. Zheng, TensorFlow: Large-scale machine learning on heterogeneous systems, software available from tensorflow.org (2015).
URL `https://www.tensorflow.org/`

[39] F. Haddadpour, M. M. Kamani, A. Mokhtari, M. Mahdavi, Federated learning with compression: Unified analysis and sharp guarantees, arXiv preprint arXiv:2007.01154 (2020).

[40] M. Yanjun, Y. Dianhai, W. Tian, W. Haifeng, Paddlepaddle: An open-source deep learning platform from industrial practice[j], Frontiers of Data and Computing 1 (1) (2019) 105–115.

[41] T. Ryffel, A. Trask, M. Dahl, B. Wagner, J. Mancuso, D. Rueckert, J. Passerat-Palmbach, A generic framework for privacy preserving deep learning (2018). `doi:10.48550/ARXIV.1811.04017`.
URL `https://arxiv.org/abs/1811.04017`

[42] I. Kholod, E. Yanaki, D. Fomichev, E. Shalugin, E. Novikova, E. Filippov, M. Nordlund, Open-source federated learning frameworks for iot: A comparative review and analysis, Sensors 21 (1) (2020) 167.

[43] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, M. Sinn, et al., Ibm federated learning: an enterprise framework white paper v0. 1, arXiv preprint arXiv:2007.10987 (2020).

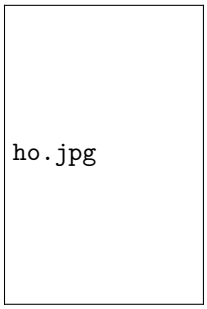
[44] L. Biewald, Experiment tracking with weights and biases, software available from wandb.com (2020).
URL `https://www.wandb.com/`

Biographies

Mohamad Arafeh

Is currently pursuing a Ph.D. degree with École de Technologie Supérieure, Montreal, QC, Canada. He is working in the area of federated learning.

Hadi Otrok



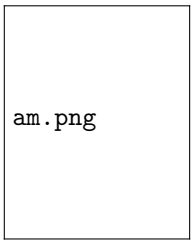
Received the Ph.D. degree in computer science and software engineering from Laval University, Canada, in 2005. He is a Professor with Concordia Institute for Information Systems Engineering, Concordia University, Canada. From 2005 to 2006, he was a Postdoctoral Fellow with Laval University, and then NSERC Postdoctoral Fellow at Simon Fraser University, Canada. He is an NSERC Co-Chair for Discovery Grant for Computer Science (2016-2018). His research interests include the areas of computational logics, model checking, multi-agent systems, services computing, game theory, and deep learning.

Hakima Ould-Slimane

Obtained her Ph.D. degree in Computer Science from Laval University, Quebec, Canada. She is currently a professor at the department of mathematics and computer science at Universite de Quebec a Trois-Rivieres (UQTR, Trois-Rivieres, Canada). Her research interests include mainly: information security, cyber re-

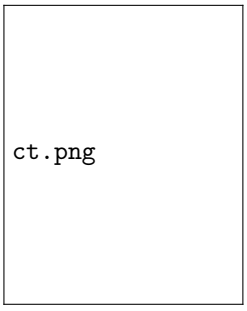
silience, homomorphic encryption, federated learning, preserving data privacy in smart environments, machine learning based intrusion detection, access control, optimization of security mechanisms and security of social networks.

Azzam Mourad



Received his M.Sc. in CS from Laval University, Canada (2003) and Ph.D. in ECE from Concordia University, Canada (2008). He is currently Professor of Computer Science and Founding Director of the Cyber Security Systems and Applied AI Research Center with the Lebanese American University, Visiting Professor of Computer Science with New York University Abu Dhabi and Affiliate Professor with the Software Engineering and IT Department, Ecole de Technologie Superieure (ETS), Montreal, Canada. His research interests include Cyber Security, Federated Machine Learning, Network and Service Optimization and Management targeting IoT and IoV, Cloud/Fog/Edge Computing, and Vehicular and Mobile Networks. He has served/serves as an associate editor for IEEE Transactions on Services Computing, IEEE Transactions on Network and Service Management, IEEE Network, IEEE Open Journal of the Communications Society, IET Quantum Communication, and IEEE Communications Letters, the General Chair of IWCMC2020, the General Co-Chair of WiMob2016, and the Track Chair, a TPC member, and a reviewer for several prestigious journals and conferences. He is an IEEE senior member.

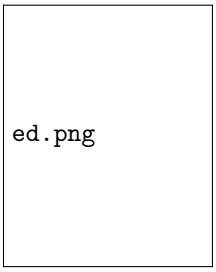
Chamseddine Talhi



Received the Ph.D. degree in computer science from Laval University, Quebec, QC, Canada, in 2007. He is a Professor with the Department of Software Engineering and IT, ÉTS, University of Quebec, Montreal, QC, Canada. He is leading a research group that investigates smartphone, embedded systems, and IoT security.

His research interests include cloud security and secure sharing of embedded systems.

Ernesto Damiani



is currently a Full Professor at the Department of Computer Science, Universita degli Studi di Milano, where he leads the Secure Service-oriented Architectures Research (SESAR) Laboratory. He is also the Founding Director of the Center for Cyber–Physical Systems, Khalifa University, United Arab Emirates. He received an Honorary Doctorate from Institute National des Sciences Appliquees de Lyon, France, in 2017, for his contributions to research and teaching on big data analytics. He is the Principal Investigator of the H2020 TOREADOR project on Big Data as a Service. He serves as Editor in Chief for IEEE Transactions on Services Computing. His research interests include cybersecurity, big data, and cloud/edge processing, and he has published over 600 peer-reviewed articles and books. He is a Distinguished Scientist of ACM and was a recipient of the 2017 Stephen Yau Award.