Journal of Internet Services and Applications, 2024, 15:1, doi: 10.5753/jisa.2024.3826

ATHENA-FL: Avoiding Statistical Heterogeneity with One-versus-All in Federated Learning

Lucas Airam C. de Souza [**Universidade Federal do Rio de Janeiro** | ***airam@gta.ufrj.br***]
Gustavo F. Camilo [**Universidade Federal do Rio de Janeiro** | ***gustavo@gta.ufrj.br***]
Gabriel Antonio F. Rebello [**Universidade Federal do Rio de Janeiro** | ***gabriel@gta.ufrj.br***]
Matteo Sammarco [**Stellantis** | ***matteo.sammarco@stellantis.com***]
Miguel Elias M. Campista [**Universidade Federal do Rio de Janeiro** | ***miguel@gta.ufrj.br***]
Luís Henrique M. K. Costa [**Universidade Federal do Rio de Janeiro** | ***luish@gta.ufrj.br***]

✉ *Centro de Tecnologia - 972, Av. Horácio Macedo, 2030 - Sala H-301 - Cidade Universitária da Universidade Federal do Rio de Janeiro, Rio de Janeiro - RJ, 21941-598, Brazil.*

Received: 9 November 2023 • **Accepted:** 14 May 2024 • **Published:** 14 August 2024

Abstract Federated learning (FL) is a distributed approach to train machine learning models without disclosing private data from participating clients to a central server. Nevertheless, FL training struggles to converge when clients have distinct data distributions, which leads to an increased training time and model prediction error. We propose ATHENA-FL, a federated learning system that considers clients with heterogeneous data distributions to generate accurate models in fewer training epochs than state-of-the-art approaches. ATHENA-FL reduces communication costs, providing an additional positive aspect for resource-constrained scenarios. ATHENA-FL mitigates data heterogeneity by introducing a preliminary step before training that clusters clients with similar data distribution. To handle that, we use the weights of a locally trained neural network used as a probe. The proposed system also uses the one-*versus*-all model to train one binary detector for each class in the cluster. Thus, clients can compose complex models combining multiple detectors. These detectors are shared with all participants through the system's database. We evaluate the clustering procedure using different layers from the neural network and verify that the last layer is sufficient to cluster the clients efficiently. The experiments show that using the last layer as input for the clustering algorithm transmits 99.68% fewer bytes to generate clusters compared to using all the neural network weights. Finally, our results show that ATHENA-FL correctly identifies samples, achieving up to 10.9% higher accuracy than traditional training. Furthermore, ATHENA-FL achieves lower training communication costs compared with MobileNet architecture, reducing the number of transmitted bytes between 25% and 97% across evaluated scenarios.

Keywords: Federated learning, non-IID data, privacy-preserving AI.

1 Introduction

Machine learning enables task automation by creating models that identify patterns in datasets to predict or classify future data. In traditional machine learning systems, model training requires client-data collection, which usually reveals private or sensitive information from the user or collection point Liu *et al.* [2021]. Also, the high data volume generated by devices imposes a challenge to this practice of centralizing information into a single point in the network. Therefore, Federated Learning (FL) has emerged as a proposal for training machine learning models that preserve the privacy of the user without sharing local data. Federated learning (FL), proposed by Google McMahan *et al.* [2017], has become popular among researchers and industry due to the possibility of training machine learning models while preserving users' data privacy de Souza *et al.* [2020]; Djenouri *et al.* [2023]; Li *et al.* [2023a]; Singh *et al.* [2022]. After the change in data processing regulations in several countries, for instance, the California Consumer Privacy Act (CCPA) in the USA and the General Data Protection Regulations (GDPR) in Europe, the importance of federated learning research and adoption increased.

Training in federated learning replaces data sharing with model parameter sharing. In Federated Averaging (FedAVG), the most widely used algorithm for model parameters' aggregation in FL, clients train the model locally for a few epochs and send the result to the aggregation server, which combines the individual trained models into a single global model. Then, the aggregation server broadcasts the global model back to the clients, which further improves it with fresh training data. The aggregation server and clients repeat this process until the global model converges or the training reaches the amount of maximum global epochs. FedAVG is a specific algorithm for FL models based on updating parameters via the loss gradient vector and averaging the vectors sent by the clients to update the global model. Thus, clients' training samples remain stored locally, preserving data and user privacy. Nevertheless, another problem persists when deploying FL on a large scale: clients have heterogeneous data generated from non-independent and identically distributed (non-IID) distributions, leading to convergence difficulty and suboptimal performance Zhao *et al.* [2018]. Therefore, a proposal that reduces the effects of data heterogeneity during model training and allows the generalization of the classifier to samples originating from other data distributions becomes necessary. Clustering clients by data similarity allows models to be trained on IID data and quickly converge with high final classification performance Ouyang *et al.* [2021].

Zhu *et al.* [2021] propose using one-*versus*-all classification scheme to mitigate the impact of heterogeneous data in training federated learning models. Their Federated OvA (*FedOVA*) algorithm uses models for binary classification and selects clients having samples of the target class to perform the training. Nevertheless, their proposal lacks a study on the impact of the model creation and an adequate protocol for detector training. ATHENA-FL groups clients according to data distributions, before training the OvA model, thus reducing detector training time.

This paper proposes ATHENA-FL (Avoiding Statistical Heterogeinety with One-*versus*-All in Federated Learning), a system that uses model training with the one-*versus*-all (OvA)[1] technique to enable model sharing between groups. ATHENA-FL reduces statistical heterogeneity by grouping clients by data similarity. The one-*versus*-all model uses independently-trained binary classifiers which estimate the probability that a sample belongs to the class identified by a detector. After training, the detectors are combined for sample classification. Each detector estimates the probability that the sample belongs to its class, and the classifier labels the sample from the detector that generates the highest probability. Thus, the one-*versus*-all method is used for efficient model sharing between groups to create a generic model for classifying data from different groups. In addition, the system groups clients according to their data distributions so the detectors converge quicker with high classification performance.

We evaluate the accuracy evolution of the detectors and compare the classification performance and communication requirements for training the models. The accuracy of the models depends on the diversity of the samples used for training the detectors, with the accuracy of the one-*versus*-all model up to 10.9% higher than the MobileNet architecture Howard *et al.* [2017]. At the same time, the amount of bytes transmitted over all training epochs is reduced by up to 97.37% using the one-*versus*-all model instead of MobileNet. Thus, the system provides an effective way to train FL models even in scenarios with heterogeneous distributions.

We summarize the contributions of this paper as follows:

- We propose a system that clusters federated learning clients to reduce data heterogeneity. Each cluster trains its models to detect specific classes.
- We generalize the models trained in each cluster by executing the One-*versus*-All classification approach. The clusters share the models to create an ensemble capable of classifying samples among different clusters.
- We evaluate the communication requirements of our proposal. The results show that we can reduce the communication requirements in two ways. First, using only the final layer to cluster the clients, which reduces the communication in the cluster assignment phase. Second, training shallow neural networks as the OvA detectors. These models converge faster, thus, they are capable of reduce the total transmitted bytes compared with deeper neural networks, maintaining almost the same accuracy level in different datasets.

This paper is organized as follows. Section 2 presents basic concepts about the Federated Learning scenario and the One-*versus*-All model. Section 3 reviews the state-of-the-art proposals to increase the classification performance of federated learning systems and reduce the impacts caused by data heterogeneity. Section 4 describes the proposed ATHENA-FL system. Section 5 presents the development of a prototype of ATHENA-FL and the analysis of the obtained results. Finally, Section 6 concludes this work and discusses future research directions.

2 Federated Learning Concepts

Figure 1 exhibits the execution diagram of the proposal presented by Google. First, the aggregation server randomly selects a set of clients that will participate in epoch n. In the example in Figure 1 clients 1 and 4 are selected. In this step, the clients calculate the new model state for epoch $n-1$ using local data. The computed models are transferred to the aggregation server, which aggregates into the epoch n global model. Among the existing aggregation algorithms, the most popular is the Federated Average (FedAVG), which calculates the weights' average in each model layer to perform the model update. Federated learning average is formulated as an optimization problem in which we search for an optimal solution that minimizes the loss function by adjusting the w model's parameter over the parameter space V^a as formulated in Equation 1. After aggregation, the global model for epoch n is generated and the aggregation server shares it with all clients. The process runs until one of the stopping conditions is met, such as the maximum number of epochs, classification performance, or convergence of classification metrics:

$$\underset{w\in V^a}{min}\left(f(w)\overset{\text{def}}{=}\sum_{n=1}^{N}p_nF_n(w)\right). \tag{1}$$

In the formulation above, N is the number of clients in the system and p_n is the neural network weights of the n-th client, a.k.a model's parameter, $p_n \leq 1$ and $\sum_n p_n = 1$. In FedAVG, $p_n = s_n/s$, where s_n is the size of the n-th client dataset and s is the number of samples among all clients. $F_n(w)$ is the local objective function for the client n. $F_n(w) = \frac{1}{s_n}\sum_{k\in s_n} f_i(w)$. The function $f_i(w)$ is related to $l(x_i, y_i; w)$, the loss function for the prediction (x_i, y_i) with model parameter's w. FL algorithms' convergence conditions depend on feature distribution among clients and data distribution. Hence, we define the classification of federated learning according to the feature distribution and discuss how the data can be distributed in the system.

[1] Available at `https://github.com/GTA-UFRJ/ATHENA-FL`

Table 1. List of acronyms used.

ATHENA-FL	Avoiding sTatistical HEterogeiNety with one-*versus*-All in Federated Learning
CCPA	California Consumer Privacy Act
CEFL	Communication-Efficient Federated Learning
CFL	Clustered Federated Learning
CIFAR	Canadian Institute For Advanced Research
DBSCAN	Density-Based Spatial Clustering of Applications with Noise
DCS	Decomposed Cosine Similarity
FedAVG	Federated Averaging
FedOVA	Federated OvA
FedTP	Federated learning by Transformer Personalization
FL	Federated Learning
FLEE	Federated Learning Early Exit of inference
FlexCFL	Flexible Clustered Federated Learning
FMNIST	Fashion MNIST
GDPR	General Data Protection Regulations
Hier-SFL	Hierarchical Split Federated Learning
IFCA	Iterative Federated Clustering Algorithm
IID	Independent and Identically Distributed
LDA	Label-based Dirichlet Partition
MNIST	Modified National Institute of Standards and Technology
OPTICS	Ordering Points to Identify the Clustering Structure
OvA	One-\textit{versus}-All
RAM	Random Access Memory
StoCFL	Stochastic Clustered Federated Learning
TDT	Total Data Transmitted

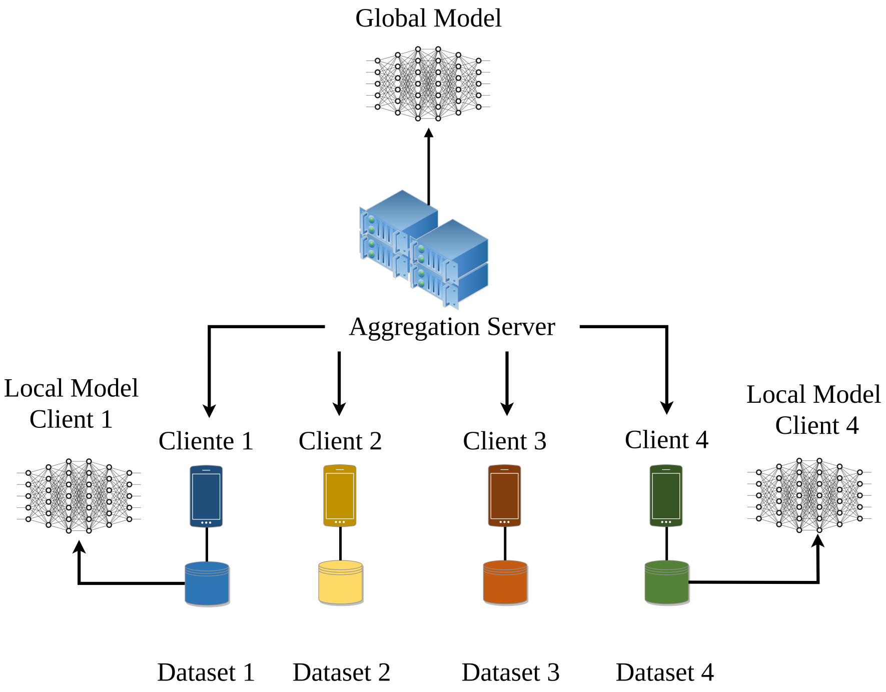


Figure 1. Federated learning architecture with four clients. Round n starts with the aggregation server randomly selecting some clients. Selected clients use their datasets to enhance the global model for epoch $n-1$ and build the local model for epoch n. Each client communicates with the aggregation server independently, and their datasets remain on their devices throughout the training process. The aggregation server receives local updates from clients and aggregates the responses to the global model, generating the global model for round n. Finally, the global model for round n is sent to clients. The process is repeated until a stopping condition is reached, such as model convergence or the maximum number of global epochs.

2.1 Feature Distribution

Federated learning has three classifications according to the features that each client has Yang *et al.* [2019]: Horizontal, vertical, or federated transfer learning.

Horizontal federated learning: In horizontal federated learning, clients have the same feature space and learning task, however, clients' samples are different. Horizontal learning has difficulty establishing previously which features the clients should collect and use in model training. However, the training and aggregation process is simple, as all clients use identical input sizes.

Vertical federated learning: Another form of federated learning is vertical, which assumes a scenario in which clients have non-intersecting feature distributions. This is the most challenging type of learning, as clients collect data that can vary in dimension, making the global model creation process more complex. However, there is no need to communicate which features are extracted, so clients have a more robust privacy model.

Transfer federated learning: This classification assumes that the participants have some similarities in the learning task. Thus, it is possible to train a model a few epochs in a dataset and afterward, execute a fine-tuning in neural network weights with the final learning task to improve the model performance.

2.2 Data Distribution

In this section, we discuss sources of non-IID data distributions. Ma *et al.* [2022] identify five scenarios where data are heterogeneous among clients: Feature distribution skew, label distribution skew, vertical federated learning, label inconsistency, and quantity skew, which we consider in this work. Below are the details and characteristics of each one:

Feature distribution skew: When a client collects data, it can be specific rather than general. Therefore, the features might significantly differ from one user to another. The data can be related to geographical locations, such as temperature data, or even to user personality, such as vocabulary and handwriting traces.

Label distribution skew: Datasets that have imbalance classes have this type of heterogeneity. For instance, client 1 has 80% samples of class 1 while the other 20% are samples of the remaining classes, and client 2 has the opposite data behavior, 20% samples are from class 1 and the 80% are distributed among the remaining classes. Our experiments consider specifically this type of heterogeneity.

Vertical federated learning: This learning scenario is inherently Non-IID since clients have different feature space distributions.

Label inconsistency: Clients may disagree with sample labels. This can occur because the parties disagree on which class a sample belongs to, or the labeling processing lacks a well-defined protocol to assign classes to samples. For example, categorical features such as good, regular, and bad might subjectively change according to the professional responsible for defining a sample class.

Quantity skew: In this scenario, clients significantly differ in the dataset size. For instance, client 1 has $|s_1| = 2,000$ while client 2 has $|s_2| = 500,000$. The difference between quantity skew and label distribution skew is that in quantity skew sample might originate from the same distribution but with different sample sizes among the clients.

2.3 One-*versus*-All Model

This model comprises a generic classifier, an ensemble of several specialized classifiers. Each classifier is trained to detect just a single class. The specialized classifiers are binary classifiers that return how similar the sample is in comparison to the samples of its class. Thus, they are also named as detectors. An advantage of this approach is the model's size since the binary classifier can be implemented as a shallow neural network. Since they are binary classifiers, they tend to converge faster in fewer epochs than a deeper neural network. Therefore, we can reduce the communication resource requirements using this approach. Equation 2 shows the classification process of a sample x:

$$C_{ova}(x) = \underset{i \in [1,R]}{\operatorname{argmax}}\, r_k(x). \tag{2}$$

The model comprises several detectors r_k, which classify the sample x and return the probability that the sample belongs to the class k. The classifier C_{ova} assigns to the sample the label that has the highest probability among all detectors.

The model must execute a pre-process to train the detectors. This pre-processing consists of transforming the original samples' labels into binary labels. Thus, the detector's training needs an auxiliary function to binarize the labels according to the current detector. The binarization function is defined as:

$$B(k) = \sum_{n \in G} \nVdash_k(s_n), \tag{3}$$

if the i-th label belongs to the k-th class, the function returns 1 and zero otherwise. Each detector can be trained separately.

We highlight two advantages of using this approach. First, detectors are smaller, hence the time between training epochs is reduced compared to deeper neural networks. Secondly, the one-*versus*-all model can identify the classes from present on other clusters easily, since we merely combine the detectors from the system's detectors database.

3 Related Work

One of the most relevant challenges regarding federated learning is increasing the final model performance while reducing the total training time. Although hyperparameter optimization is a promising alternative Neto *et al.* [2022], client selection Fu *et al.* [2022] or local model personalization Tan *et al.* [2022] could also achieve faster convergence. While the initial federated learning proposal considers a uniform distribution for the probability that a client participates in training McMahan *et al.* [2017], client selection modifies this likelihood function to reflect the client's ability to significantly contribute to model training in a global epoch. On the other hand, model personalization locally adjusts models after the overall training has reached a predefined performance threshold. In this way, client selection increases global model performance, while personalization increases local performance. Another strategy of model personalization is clustered FL, where the idea is to divide clients into clusters to train models with similar data and achieve higher accuracy.

3.1 Selecting Clients for Efficient Training

Luo *et al.* [2022] and Lai *et al.* [2021] propose client selection schemes that optimize model convergence speed in federated learning environments. Selecting clients based solely on data representativeness decreases the total number of epochs for model convergence. Nevertheless, clients that have more relevant data for the problem might have a longer training time for each epoch, according to the amount of data and characteristics of the hardware used. On the one hand, increasing the time between epochs implies a higher overall delay, as federated learning environments are generally synchronous and wait for all clients to respond or time out. On the other hand, selecting clients with higher computational power to reduce the time between epochs may incur more epochs for convergence if the selected clients have statistically insignificant data for the learning task. There is a trade-off between the number of epochs for model training and the total time for each epoch. Thus, the authors simultaneously consider the devices' characteristics and their data distribution relevance to reduce the convergence time of the global model. While Luo *et al.* propose a non-convex function regarding the expected number of epochs based on clients' previous updates, Lai *et al.* avoids the linear programming, creating a selection function that evaluates clients' training losses.

Wang *et al.* [2020a] propose to apply reinforcement learning for client selection. The reinforcement learning agent is tuned to select the best clients for each training epoch and provide the best subset of clients given the current global model state. The system estimates clients' statistical relevance through the loss gradient vector sent to the aggregation server. This procedure maintains the privacy of the client's data. Fu *et al.* [2022] present a survey of client selection systems and frameworks in federated learning. Their paper presents the state-of-the-art in client selection and compares their main differences, demonstrating that client selection has great potential to produce more accurate models in federated learning with less training time.

FedProx is a framework to reduce the effects of statistical and system heterogeneity Li *et al.* [2020]. The difference between FedProx and FedAVG is the flexible amount of work the clients perform. While FedAVG distributes an equal amount of work to all clients, FedProx considers individual hardware constraints to assign work to clients proportionally. Also, the authors propose inserting a regularization term on the local objective function to minimize the impact of non-IID data in the federated network.

Nishio and Yonetani [2019] propose a protocol for client selection in federated learning in which clients with higher processing capabilities and lower communication latency are prioritized in the presented selection scheme. Liu *et al.* [2020] propose a hierarchical architecture for federated learning, which performs local aggregations on clients, partial aggregation at the network edges, and global convergence in the cloud. Processing data at the edge decreases communication latency, but the restricted number of devices can make it difficult for the global model to converge. Cloud processing, however, can access more devices and capture more data variance at the cost of higher communication latency. Thus, the authors apply federated training on levels: client-edge and edge-cloud. Clients maintain low latency for communication with the edge, and the cloud can access the models generated at the edges. Hierarchical Split Federated Learning (Hier-SFL) Qin *et al.* [2023] is an FL framework that divides the neural network into three different levels, client-edge-cloud, to train the model. Thus, each one of the neural network parts has an associated loss function, which is adjusted separately. Hier-SFL objective is to reduce the communication overhead during the training. However, the distribution of data generated by clients is not considered, which can affect the performance of models in scenarios with non-IID distributions.

Rai *et al.* [2022] propose the irrelevance sampling metric for client selection to improve the final accuracy of federated learning models in IID and non-IID scenarios. The objective is to select clients considering the quality and quantity of their samples. Each client computes its irrelevance sampling metric and sends it to the server. The server then clusters the clients according to the informed value in the following three clusters: positive, negative, and zero. Finally, the proposed methodology randomly selects the clients of each cluster to achieve faster model convergence.

Fraboni *et al.* [2021] propose a clustering sampling method for client selection. The authors provide two approaches for clustering clients: client sample size and model similarity. Nevertheless, the first approach highly depends on the clients' informing their exact sample size to the aggregation server for cluster definition. Therefore, this approach is susceptible to clients' malicious behaviors. On the other hand, the method uses all model weights in the model similarity proposal, which introduces communication overhead.

The selection of clients decreases the time to convergence and increases the final accuracy of the model. The client selection, however, is insufficient for practical purposes in environments with heterogeneous data. Thus, an alternative is model personalization, which fine-tunes clients' models according to their local data and improves their final performance.

3.2 Personalized Models in FL

FedTP (*Federated learning by Transformer Personalization*) is a framework for reducing data heterogeneity by transforming datasets and personalizing models Li *et al.* [2023b]. The objective of FedTP is to define the basis of a transformation. In this transformation, the client data is similar, and they can personalize their models in some layers overcoming the convergence problems. However, the proposal generates significant communication overhead, requiring a long time to adjust the projection parameters and for the model to converge.

FLEE (*Federated Learning Early Exit of inference*) Zhong *et al.* [2022] is a hierarchical-federated learning framework that splits the model into three different geolocations. The model's division between the cloud, edge, and end device allows using the method of early exit inference from neural networks. Furthermore, the hierarchical division of the training reduces the impact caused by non-IID data distributions on the model convergence, as the authors assume that the data have higher similarity according to the geographic distance of the clients.

Ditto Li *et al.* [2021] is a framework for personalized federated learning. The authors separate the optimization problem into two parts: global optimization and local optimization. Global optimization considers the contribution of all clients to the model, while local optimization regularizes the client's model according to the local data. Also, the authors provide two definitions: robustness and fairness in federated learning environments. Robustness expresses the model's capacity to achieve high accuracy even in heterogeneous data or data poisoning attacks. Fairness represents how different clients' performance is when using local test data. We use these definitions to evaluate our proposal.

Table 2. Comparison of related works

Work	Heterogeneity Reduction	Strategy Algorithm	Strategy Input	Iterative	Hierarchy
Ouyang *et al.* [2021]	Client Clustering	Periodic clustering	All models' weights	YES	1
Wang *et al.* [2020a]	Client Selection	Reinforcement learning	All Weights	NO	0
Luo *et al.* [2022]	Client Selection	Optimization	Users' resources	NO	0
Lai *et al.* [2021]	Client Selection	Client's utility measurement	Users' resources	NO	0
Li *et al.* [2020]	Client Selection	Model regularization	Proximal term value	NO	0
Nishio and Yonetani [2019]	Client Selection	Optimization	Users' resources	YES	0
Liu *et al.* [2020]	Hierarchical Training	Geographical training	Users' Location	NO	2
Qin *et al.* [2023]	Hierarchical Training	Hierarchical early-exit	Transformed Data	YES	2
Rai *et al.* [2022]	Client Selection	Irrelevance measurement	Irrelevance Metric	NO	0
Fraboni *et al.* [2021]	Client Selection	Clustered sampling	Sample size or similarity	NO	0
Li *et al.* [2023b]	Model Personalization	Learn-to-personalize	Local data	NO	0
Zhong *et al.* [2022]	Hierarchical Training	Hierarchical early-exit	Users' Location	NO	2
Li *et al.* [2021]	Model Personalization	Two-steps optimization	All Weights	YES	2
Dennis *et al.* [2021]	Client Clustering	Distributed clustering	Users' centroid	NO	1
Zhu *et al.* [2021]	Model Personalization	One-versus-All	Users' response	NO	0
Chu *et al.* [2022]	Client Clustering	Louvain method	Euclidean distance	NO	1
Zeng *et al.* [2023]	Client Clustering	Cosine similarity	Users' loss function	NO	1
Ghosh *et al.* [2020]	Client Clustering	Iterative clustering	Clients' decision	YES	Multiple
Duan *et al.* [2022]	Client Clustering	DCS	All Weights	YES	1
Sattler *et al.* [2020]	Client Clustering	Recursive clustering	Loss gradient vector	YES	Multiple
Our	Client Clustering	Generic clustering	Last Layer Weights	NO	1

3.3 Clustered Federated Learning

Clustered federated learning is a subfield of model personalization in federated learning research. This subfield focuses on creating strategies to arrange FL clients into groups, reducing data heterogeneity. The proposals are mostly differentiated by clustering strategy, metrics, and training.

CEFL (*Communication-Efficient Federated Learning*) is an FL-based framework for medical-data model training Chu *et al.* [2022]. The authors determine the similarity between clients by calculating the Euclidean distance of the clients' neural network weights. Based on similarity, the system clusters clients using the Louvain method Blondel *et al.* [2008]. In each group, the client with the highest sum of similarity is the leader. The leader performs federated training of the model's first layers with leaders from other groups. The model's final layers are trained individually in each group.

StoCFL (*Stochastic Clustered Federated Learning*) Zeng *et al.* [2023] is a federated learning framework that clusters clients according to the cosine similarity. The proposal introduces two models: the global model and the cluster model. The global model maintains information from all clusters, while clients only participate in model training inside the cluster they are involved. Meanwhile, there is a high cost to the system's clients, who must simultaneously train the two models.

IFCA (Iterative Federated Clustering Algorithm) Ghosh *et al.* [2020] is a proposal for clustering clients to personalize their models. In the proposal, clients are responsible for choosing their clusters. In addition, the authors propose the use of multi-task learning, which consists of sharing some neural network weights for clients who have data distributions with intersections but are in different clusters. Nevertheless, the downside of delegating the process of identifying groups to the clients is that the environment becomes susceptible to malicious behavior and requires more computational costs from the clients' devices. Another disadvantage of this proposal is to assume that the number of groups is known a priori, which may be unfeasible and can either overestimate or underestimate the number of existing clusters.

CFL Sattler *et al.* [2020] recursively partitions federated learning clients into more homogeneous groups. This procedure mitigates the problems generated by non-IID distributions. Partitioning occurs whenever the loss gradient vector exceeds a pre-established distance threshold. However, clients' recursive partition leads to computational overhead on the aggregation server since it needs to run the procedure at each global training epoch. ClusterFl Ouyang *et al.* [2021] is a framework for creating homogeneous clients' clusters in federated learning environments. The authors propose clusters' periodic verification to detect inefficient clients, aiming to reduce the convergence time for the cluster model. However, the proposal may present unnecessary high computational overhead, as the authors foresee models' periodic retraining. Furthermore, CFL and ClusterFl proposals do not combine information from models of other clusters, which might reduce the accuracy for clients with generic tasks.

FlexCFL Duan *et al.* [2022] is a framework that considers clients' data distribution time shifts. The grouping strategy is static, which avoids rescheduling clients for each epoch as Sattler *et al.* [2020] and Ghosh *et al.* [2020] do. Also, the framework uses only a subset of clients to determine the number of clusters in the system. The remaining clients are clustered after the decision about the number of clusters. Before each round, the authors execute a strategy to detect if the clients remain with the same distribution or if it has changed. When the distance exceeds a predefined threshold the client needs to perform the clustering step again. Nevertheless, the proposal requires higher computational resources than our proposal to detect the data shift.

de Souza *et al.* [2023a] propose a cluster federated learning system in which each cluster trains a single model. Clients use FedAVG to train the cluster model after the clustering step. This approach increases the final model's accuracy and reduces the time to convergence. However, the proposal is limited to specific tasks, like classifying data similar to the cluster training data. Thus, clients only retain local cluster knowledge. Our current work improves the system, providing a way to combine the models generated in different clusters through the One-versus-All model.

We propose ATHENA-FL, a federated learning system based on clustering clients by data distribution similarity and training neural network models through the One-*versus*-All (OvA) approach. Initially, our proposal clusters clients using their neural network weights as input for the clustering algorithm. Like other CFL proposals, the clustering process increases the accuracy and reduces the training time of the models of each cluster, as it makes the training data more homogeneous. Unlike previous proposals, each group trains multiple models, which are detectors of the classes present in the clients' datasets. At the end of training the models of each group, it is possible to combine them to detect the class of samples outside the cluster by using the OvA model. The performance of ATHENA-FL is compared to FedAVG using the MobileNet Howard *et al.* [2017] and MobileNetV2 Sandler *et al.* [2018] architecture, lightweight deep neural networks for image classification. Furthermore, the experiments compare the communication cost of the two approaches, calculated through the number of bytes transmitted per client during training. We show that there is a trade-off between the number of existing classes and the accuracy of the one-*versus*-all model when the clusters have few classes. More classes in the system imply more detectors, which must be trained with more diverse data for a better distinction between classes. Thus, detectors trained with few classes present a lower performance on out-of-group data, reducing the accuracy of the OvA model. This paper extends a Portuguese-written paper previously published de Souza *et al.* [2023b] by executing new experiments about the communication impact. We consider the transmission time among epochs and how the number of layers used to cluster impacts the system. Also, we show that our proposal of using just the final layer significantly reduces the input size to define clusters. Finally, we add an experiment considering a distribution generated by the LDA function, which is a more realistic form to generate non-IID data.

Table 2 summarizes the characteristics and comparisons of all related works. We classify the works due to the strategy to reduce heterogeneity, the algorithm used, the algorithm input, whether the approach is iterative, and how many hierarchical levels there are. The strategies to reduce training heterogeneity are client selection, hierarchical training, model personalization, or client clustering. Besides, there are multiple algorithms used, which depend on different inputs. The algorithm also determines if the strategy is iterative. Regarding the clustering level, 0 means that the proposal uses a single global model, and 1 the proposal has clusters where each one has a cluster global model. Multiple or 2 clustering levels mean that the cluster can be divided into one or more sub-clusters.

4 The ATHENA-FL System

ATHENA-FL is a federated learning system for training models ensuring data privacy. Our system relies on a set of clients clustered according to data similarity. Client clustering allows the training of machine learning models more efficiently than the initial federated learning proposal McMahan *et al.* [2017] under non-IID data distributions, increasing the final accuracy and reducing the total epochs for model convergence. Nevertheless, clustering creates specific models, which usually can only classify samples generated from the same cluster. Thus, we propose the adoption of the one-*versus*-all (OvA) model to share models trained on different clusters and to create generic models. The OvA learning model consists of training binary detectors for each data label. Each detector gives the probability that a sample belongs to its class. Since they execute a binary classification, their training process is easier and converges faster than a deep neural network for multi-class classification. We combine the detectors and take the one that provides the greatest probability to assign a sample predicted label. Figure 2 displays the steps proposed for executing the system.

4.1 Problem Formulation

We assume a federated learning scenario with horizontal data distribution that the clusters are formed homogeneously, therefore, clients have similar samples of the same classes. The group trains detectors for all existing classes in the client datasets. In addition, the server together with the clients uniquely identifies existing classification tasks and labels. In this way, new classes can be incorporated into the system without loss of generality in relation to existing models. Furthermore, there are no equal labels for samples belonging to different classes in the system before grouping. Therefore, our non-IID tests are based on the label distribution skew scenarios.

We cluster clients based on their data similarity to reduce non-IID effects during the training. ATHENA-FL uses the weights of the last layer as input for a clustering algorithm, which assigns clients to one of the G clusters.

ATHENA-FL uses one detector per class. Thus, instead of minimizing a single objective function, our system minimizes the classification error for each detector r_k by adjusting the parameter w_k:

$$\underset{w_k \in V^a}{min} \left(r_k(w_k) \overset{\text{def}}{=} \sum_{n=1}^{N} p_{k_n} F_{k_n}(w_k) \right), \tag{4}$$

where $r_k(w_k)$ is the detector of the k-th class, p_{k_n} is the weight of the n-th client during the k-th detector's training. Finally, $F_{k_n}(w_k)$ is the local objective for the n-th client to the k-th detector.

4.2 Data Similarity-based Clients Clustering

Initially, the system arrange clients into clusters according to the data similarity. This step produces clusters in which the training data are independent and identically distributed. IID datasets facilitate model convergence and increase final classification performance Wang *et al.* [2020b]. Nevertheless, to maintain the privacy assumptions of Federated Learning, ATHENA-FL indirectly obtains from clients' data distribution for client clustering. Thus, we use a test model, in which clients execute a few training epochs with their local data. The aggregation server uses these model weights as input for

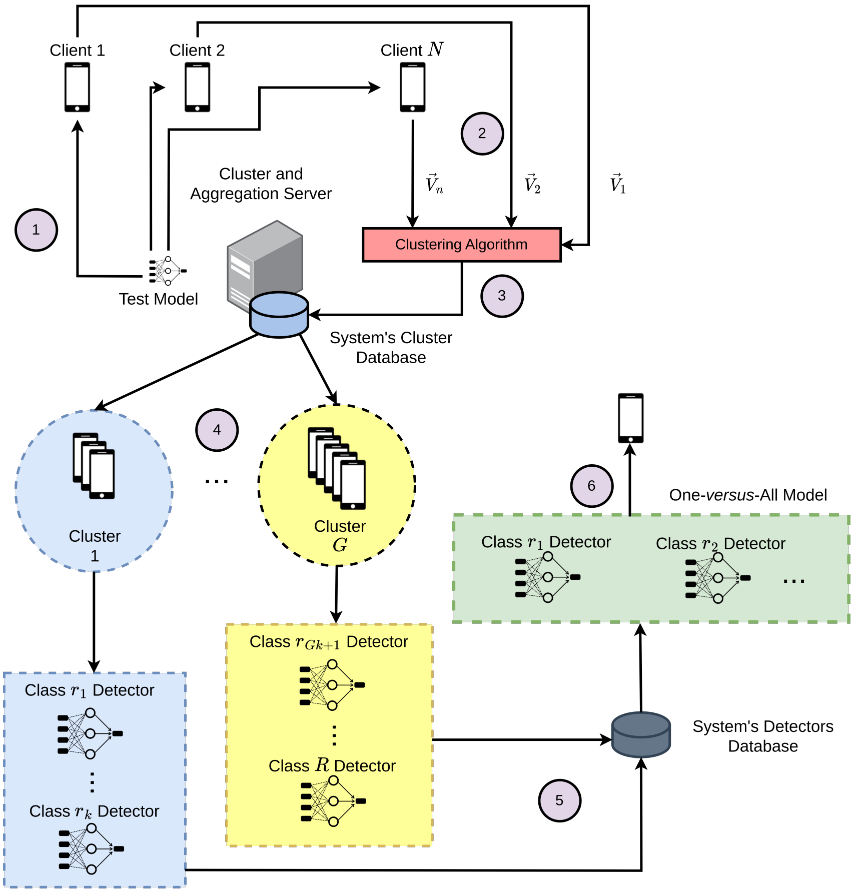


Figure 2. The ATHENA-FL execution scheme consists of 6 steps. Steps 1 to 3 mitigate the system's statistical heterogeneity by clustering clients according to the similarity of the data. In Step 4, clients train the models inside the cluster, while Step 5 shares detectors among other clusters. Finally, in Step 6, a client can combine models from different clusters to create a generic OvA model.

the Density-Based Spatial Clustering of Applications with Noise (DBSCAN) clustering algorithm to identify the different data distributions and assign clients to their clusters.

Figure 2 shows the steps executed by ATHENA-FL. Firstly, Steps 1 to 4 group clients according to data similarity. Initially, the system broadcasts a generic test model to all the clients. Then, clients adjust the neural network's weights, which depend on the private data used. After training locally, each client holds the same neural network structure, but with different weights. The clients share neural network weights with the aggregation server in Step 2. As the neural network weights are correlated with the local data, users with similar data have closer weight vectors than users with different data distributions. Therefore, they are assigned to the same cluster in Step 3. We use the DBSCAN Ester *et al.* [1996], to identify the clusters as done in a previous work de Souza *et al.* [2023a]. Unlike detectors, the model used for testing is a deep model for multiclass classification, as it is necessary to compare data distributions across all clients at this step. The clustering process ends with Step 4, which trains the detectors. New clients, absent in the group creation stage, must request the server to be allocated into a cluster by executing the previous steps.

4.3 From Specific to Generic Models

Training starts with the selection and aggregation server determining which detector will be trained federated into a group. After defining the class of interest to be trained, the server sends this information so that clients can pre-process the labels. Thus, in Step 4 of Figure 2 the system trains detectors for each class. Detector models are trained independently, concerning models from different clusters and detectors trained within each cluster. Thus, this step can be parallelized to reduce the models' training time. Each group has a total of k detectors, which can vary according to the group, and the system has R detectors in total.

The convergence time for each detector is reduced due to the clients' clustering that has approximately IID datasets. However, the models generated in the group are specific and detect only the existing classes that belong to the cluster. Thus, after its convergence, each detector is made available in the detector's database of the system in Step 5 of Figure 2. This allows clients from other groups to build one-*versus*-all models with detectors generated in the whole system.

Step 6 allows the creation of a generic model, which uses detectors trained in other clusters. After sharing models in the ATHENA-FL's detectors database, clients select the detectors of interest to create the one-*versus*-all model, C_{ova}. ATHENA-FL allows a parallel classification process, given that, as detectors are independent of each other, each model can simultaneously generate the sample's probability of belonging to the detector's class.

5 Development of the Prototype and Experimental Results

We developed the ATHENA-FL prototype using the Python v3.9.1 programming language with the Flower v1.1.0 Beutel *et al.* [2020] framework for building the federated learning environment and the Scikit-learn v1.0.2 library for creating clustering models and the Keras v2.6.0 library to create the deep learning models. The experiments were executed on an Intel Xeon CPU E5-2650 2.00 GHz server with 32 processing cores and 504 GB of RAM. We show experimental results of the models' evaluation, with the average accuracy obtained among all the clients and a 95% confidence interval. We compare our proposal with FedAVG because other approaches use different models or datasets. Thus, FedAVG establishes a baseline comparison with the state-of-the-art algorithms.

The federated learning scenario has 50 clients that train the model during 5 local epochs for a total of 200 global epochs, with a selection probability of 20%. The selection probability is the percentage of clients selected within each cluster for training global epoch. We use a batch size of 32 samples, and the accuracy results are obtained from the selection of all clients at the end of each global epoch. The clustering algorithm is defined with the distance parameter $d = 0.0279$ and the minimum number of clients per group equal to 2.

The neural network architecture used in the one-*versus*-all model was adapted from a dog and cat image identification problem, establishing a simple model for binary classification Sanguineti [2021]. Then, this neural network architecture was adapted to be the basis of the detectors used in the one-*versus*-all model due to its small size and its high capacity to correctly identify samples.

The deep architectures used in classification tasks with multiple classes are MobileNet and MobileNetV2. MobileNetV2 was chosen because it is the architecture used in the classification example of the CIFAR-10 dataset in the Flower framework and MobileNet because it has a shorter inference time and it is shallower than MobileNetV2.

We evaluate the training and performance of models on three image datasets: CIFAR-10 Krizhevsky *et al.* [2014], MNIST LeCun *et al.* [2010], and Fashion-MNIST (FMNIST) Xiao *et al.* [2017]. The CIFAR-10 dataset has 60,000 samples and a total of 10 classes representing objects or animals. The images are colored with 3 matrix which represents RGB channels and have 32x32 pixels each. The second dataset, MNIST, has 70,000 samples divided into 10 classes, which represent handwritten decimal digits. FMNIST also has 10 classes and 70,000 samples that represent different fashion clothes, divided into 60,000 for training and 10,000 for testing. MNIST and FMNIST images are provided in grayscale, originally 28x28 pixels. We process the datasets to extend to 32x32 pixels to use the same neural network architecture in all data. The datasets are equally balanced among existing classes.

We conduct four evaluations: (i) we study the impact of using different layers during the clustering step, (ii) we verify the detectors' accuracy and compare it with a complex model trained with FedAVG; (iii) we define the number of epochs necessary for both models to converge, and finally, (iv) we estimate the total amount of bytes transmitted during the training process for the two alternatives. Evaluation (i) is discussed in Section 5.1, then Section 5.2 presents the discussion of evaluations (ii) and (iii), while the last one is presented in Section 5.3.

5.1 Test Model Size Evaluation

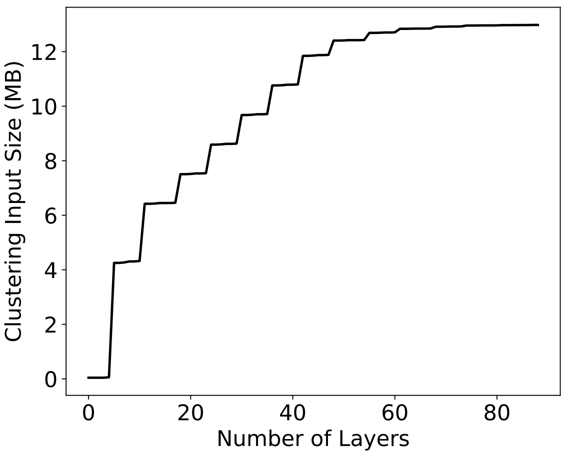
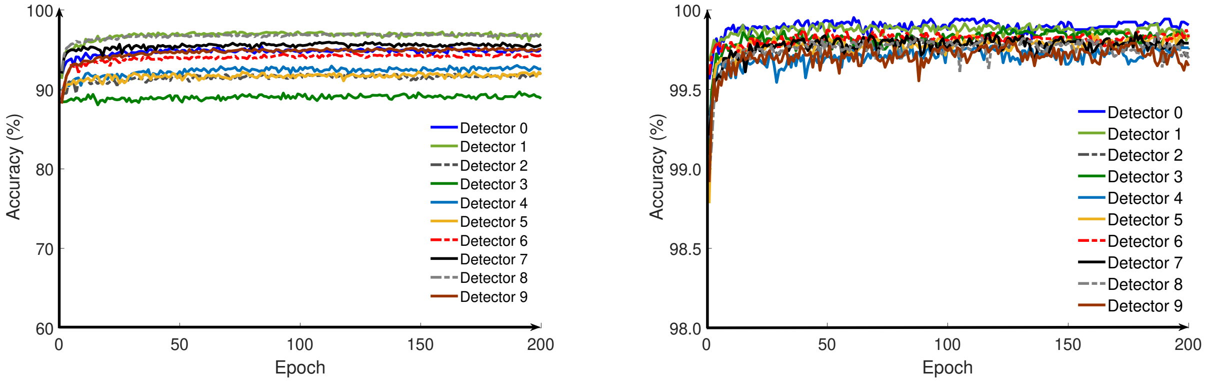


Figure 3. Amount of bytes transmitted according to the number of neural-network-layers used during the clustering process.

The goal of this experiment is to determine the amount of information needed to classify the clients into clusters better. We want to prove that using the last layer of the neural network is sufficient to cluster the clients efficiently. Thus, we execute two procedures to verify the impact on the communication of using the entire neural network. The first procedure determines how many bytes we need to transmit as a function of the number of layers used in the clustering. Secondly, we evaluate the success rate according to the number of layers used. The experiment determines the clustering efficiency using the scenario 2 classes per client as a ground truth to the KMeans number of expected clusters.

Figure 3 shows the results for the first procedure. We observe that sharing all neural network weights is required for each client 12.98 MB. Thus, it is essential to reduce the amount of transmitted bytes for a scalable scenario. Then, we evaluate different layers as the clustering input. To define a layer, we use the non-IID scenario with 2 classes per client as the ground truth for the KMeans clustering algorithm. Executing this procedure for each layer in the neural network, we observe that the final layers have more correlation with the data than the initial layers, since using the layers near the output all clients are correctly assigned to the expected clusters.

The evaluation also demonstrates that the last layer has only 41.48 kB. Therefore, using the last layer as input for the clustering algorithm reduces 99.68% of the transmitted bytes to generate the clusters.



(a) Evolution of detector's test accuracy over training epochs using the CIFAR-10 dataset.

(b) Accuracy of detector's over training epochs using the MNIST dataset.

Figure 4. Evaluation detector's accuracy when clients have IID data distributions.

5.2 Accuracy Evaluation

We evaluate ATHENA-FL on IID and non-IID data distributions. The non-IID data distributions are considered in two scenarios, in the first one each client has samples of only two classes of the dataset, and the second considers clients with samples of five different classes.

5.2.1 IID Data Scenario

The detectors quickly converge to a high accuracy value, as shown in Figure 4(a), and the final performance of the one-*versus*-all model for the CIFAR-10 dataset is $(95.5 \pm 0.3)\%$. Meanwhile, the architecture MobileNetV2 has a final accuracy of $(73 \pm 1)\%$ and the MobileNet has $(66 \pm 2)\%$. In the MNIST dataset, the performance is better, with $(99.3 \pm 0.1)\%$ and $(99.8 \pm 0.1)\%$ for MobileNet and ATHENA-FL, respectively. However, MobileNetV2 is too complex for this data and has an overfitting problem, leading to only $(7.69 \pm 0.01)\%$ accuracy. Finally, the FMNIST shows that MobileNetV2 has the best performance in the IID setting, with $(91 \pm 1)\%$ against only $(8 \pm 1)\%$ for the MobileNet and $(67.0 \pm 0.5)\%$ for ATHENA-FL, as shown in Figure 5.

Thus, the experiment shows that under IID settings, ATHENA-FL has competitive results, but in some scenarios, the detectors might need to be well-adjusted to have a better performance. The variation in performance between the datasets is due to the difficulty of the problem presented by each one. MNIST has simpler grayscale images, which are simpler for classification. Therefore, the detectors have higher accuracy and less variance in this dataset than in CIFAR-10, which has color images with more elements. Finally, the shapes of clothes in the FMNIST dataset are difficult to distinguish. Thus, we need more complex models to differentiate them. This behavior is also observed in the other evaluated scenarios.

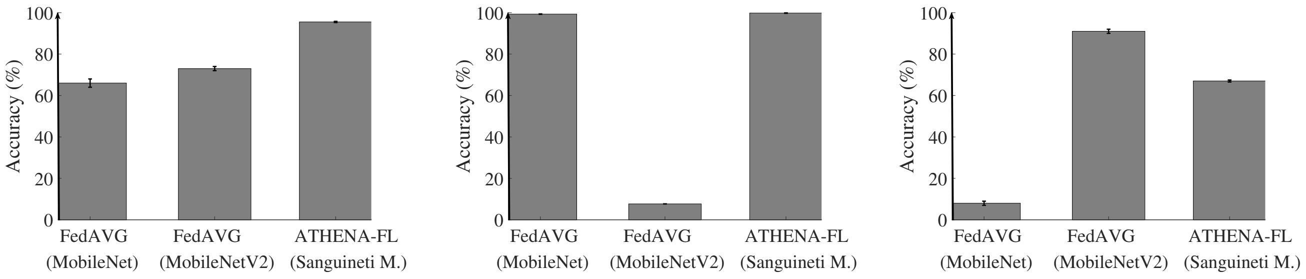
5.2.2 Scenarios with Non-IID Data

Scenarios with non-IID data consider distributions of data where clients own only a subset of the classes of the dataset. In the first non-IID case, clients have samples from five distinct classes. Figure 7 exhibits the accuracy evolution of each classifier among training epochs and Figure 6 shows the final performance of the one-*versus*-all model, MobileNet and MobileNetV2. ATHENA-FL provides the best accuracy results for all datasets in this setting, having with the CIFAR-10 dataset 10% higher accuracy compared to the MobileNet, which is the second-best model, and 10.9% higher accuracy in the MNIST dataset. Thus, the experiment demonstrates that ATHENA-FL has the potential to increase the classification accuracy up to 10.9% under the Non-IID setting compared to the MobileNet model trained purely with FedAVG.

The last scenario considers a Non-IID data distribution with two classes per client. Figure 8 shows the evolution of detectors accuracy among the training epochs for CIFAR-10 and MNIST datasets, while Figure 9 demonstrates the performance of the classifiers combined. The accuracy of ATHENA-FL was $(47 \pm 1)\%$ for the MNIST dataset, while the deep models achieved an accuracy of $(40 \pm 10)\%$ and $(10 \pm 5)\%$ for MobileNet and MobileNetV2, respectively. For the CIFAR-10 dataset, the observed accuracy is $(30 \pm 3)\%$ combining the detectors, while the MobileNet and MobileNetV2 architectures have a final accuracy of $(20 \pm 10)\%$ in this configuration. Lastly, MobileNet has $(22 \pm 2)\%$, MobileNetV2 has $(60 \pm 20)\%$, and ATHENA-FL has $(50.0 \pm 0.7)\%$ accuracy in the FMNIST dataset.

The results show that when the detectors are trained on datasets with more classes, the final classification performance is better since they can learn more patterns of the whole data distribution. This behavior can be explained by the higher variance of data from classes that are not of the detector's class. The greater variance of data in other classes allows the detector to identify more relevant features in the data of interest, instead of just differentiating specific image features that are not representative of the problem. For instance, the detectors trained in the MNIST dataset with only classes 2 and 3 have trouble differentiating 5 and 8 which have similar shapes. Nonetheless, we see that in MNIST and CIFAR-10, ATHENA-FL was able to reach up to 7% and 10% accuracy compared to the best deeper model.

Finally, our last experiment evaluates the accuracy of ATHENA-FL compared with the traditional FL approach in a non-iid scenario where the clients' datasets are not limited to samples of a subset of classes. This configuration offers

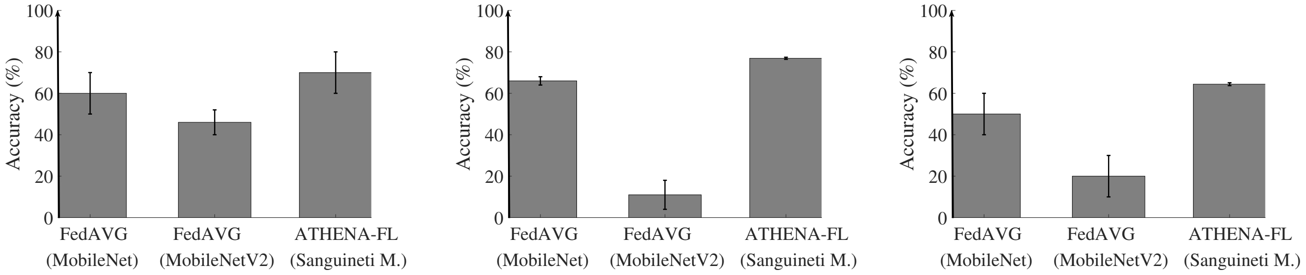


(a) CIFAR-10 dataset.

(b) MNIST dataset.

(c) FMNIST dataset.

Figure 5. Models' accuracy evaluation in the IID setting.

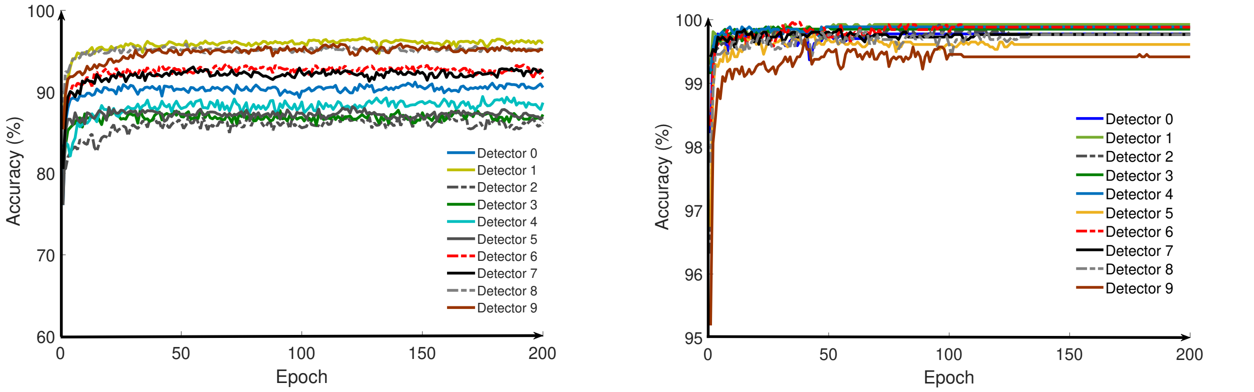


(a) CIFAR-10 dataset.

(b) MNIST dataset.

(c) FMNIST dataset.

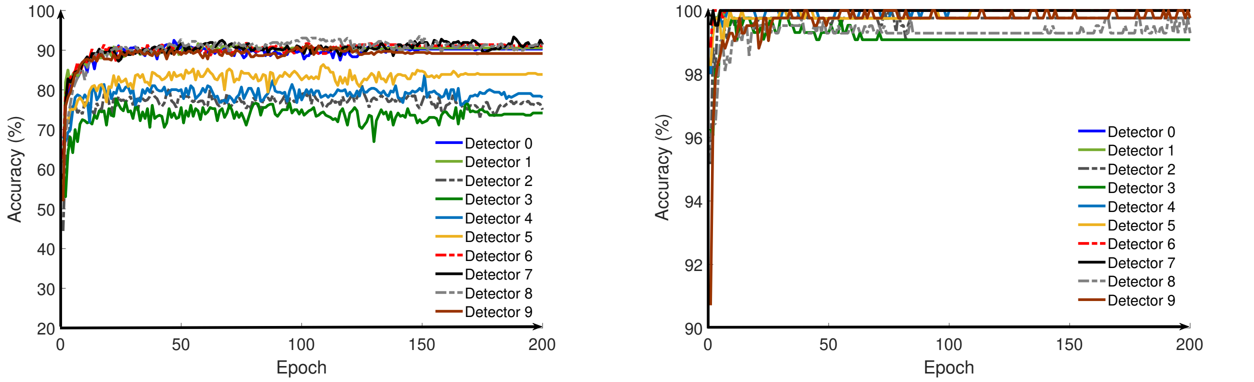
Figure 6. Final models' accuracy in federated learning with Non-IID distribution of 5 classes per client.



(a) Evolution of detector's test accuracy over training epochs using the CIFAR-10 dataset.

(b) Accuracy of detector's over training epochs using the MNIST dataset.

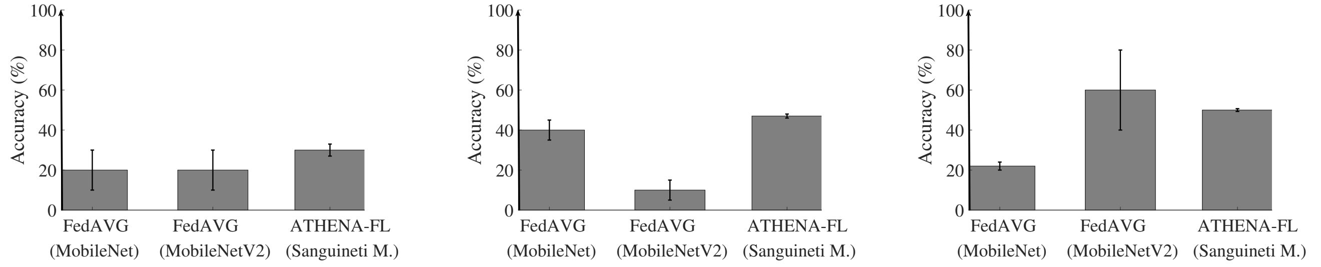
Figure 7. Evaluation detector's accuracy when clients have non-IID data distributions with 5 classes per client.



(a) Evolution of detector's test accuracy over training epochs using the CIFAR-10 dataset.

(b) Accuracy of detector's over training epochs using the MNIST dataset.

Figure 8. Evaluation detector's accuracy when clients have non-IID data distributions with 2 classes per client.



(a) CIFAR-10 dataset.

(b) MNIST dataset.

(c) FMNIST dataset.

Figure 9. Final models' accuracy in federated learning with Non-IID distribution of 2 classes per client.

a scenario closer to those encountered in real-life. We generate the dataset of each client using the Label-based Dirichlet Partition (LDA) Tang *et al.* [2021]. ATHENA-FL increases the traditional FL accuracy by almost 7×, as depicted in Figure 10.

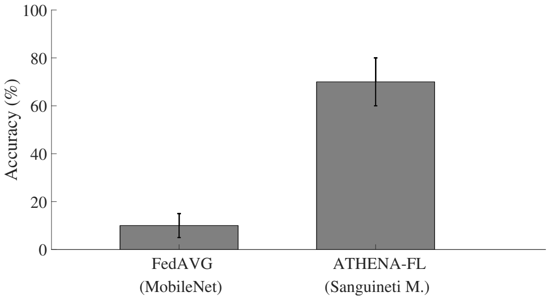


Figure 10. Final accuracy of models trained with LDA distribution in the FMNIST dataset.

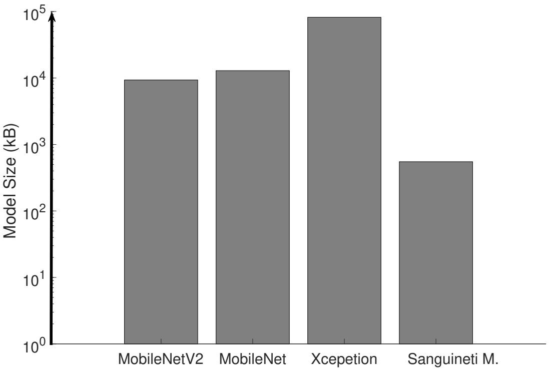


Figure 11. Comparison between the size in bytes of different neural network architectures. The size of the model directly depends on the number of parameters used by the architecture.

5.3 Communication Evaluation

The objective of this experiment is to evaluate the cost of communication between the clients and the aggregation server while training the one-*versus*-all models and a deep neural network to classify multiple classes. The communication evaluation considers the total number of bytes transmitted on average to perform model training. Let T_{dec} be the size in bytes of the detector, e_{dec} be the probability density function that indicates the number of epochs necessary for the detector to converge, and R the number of existing detectors in the problem of classification, the average amount of bytes transmitted per client $\overline{B_{ova}}$ can be expressed by the following relation, for the one-versus-all learning model:

$$\overline{B_{ova}} = T_{dec} \times R \times \mathbb{E}[e_{dec}]. \tag{5}$$

On the other hand, the communication cost of neural network architectures for multi-class classification B_{cmc} is given by:

$$\overline{B_{cmc}} = T_{cmc} \times \mathbb{E}[e_{cmc}], \tag{6}$$

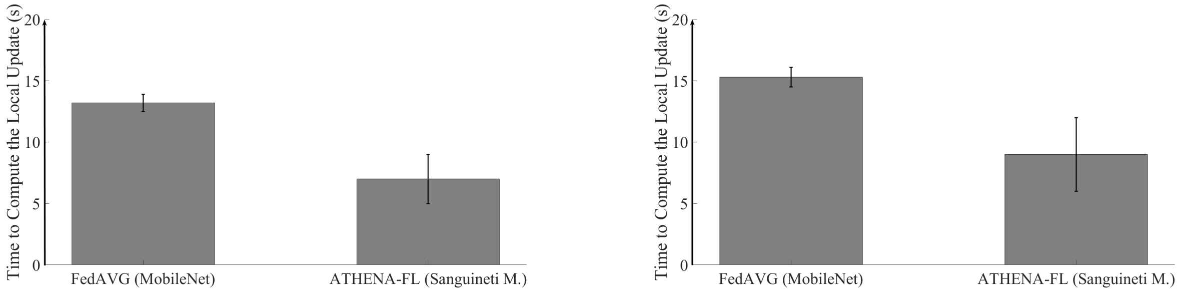
where T_{cmc} is the size and e_{cmc} is the probability density function that indicates the number of epochs required for the convergence of the multiclass classification model. The values of T_{dec} and T_{cmc} are deterministic and depend on the neural network architecture used.

Figure 11 shows the comparison between the size in bytes of the different neural network architectures evaluated in this work. Using the Keras library to create the models, each detector has $T_{dec} = 551$k bytes. Nonetheless, the MobileNetV2 architecture has 9.2MB, while MobileNet and Xception have 13MB and 81MB, respectively. The number of detectors is also a deterministic value that depends only on the dataset used for evaluation, which in the tested cases has the value $R = 10$. Furthermore, the expected values of e_{dec} and e_{cmc}, $\mathbb{E}[e_{dec}]$ and $\mathbb{E}[e_{cmc}]$, are obtained experimentally, through training performance along the epochs. When the model does not show a significant improvement concerning previous epochs, the training is considered finished. Analyzing the costs presented, so that the communication cost of the one-*versus*-all model is lower or equivalent to the multiclass neural network model, it is necessary that the total amount of bytes transmitted for training the models are related as follows: $\overline{B_{ova}} \leq \overline{B_{cmc}}$. To verify the validity of the relationship and thus the communication efficiency of the one-*versus*-all model, it is necessary to estimate the values of $\mathbb{E}[e_{dec}]$ and $\mathbb{E}[e_{cmc}]$.

The criterion to determine the epoch at which the model converged consists of comparing the accuracy in the current epoch with the accuracy in the previous epoch. If the epoch is in the limit of $tol = 0.001\%$ the previous epoch, we consider that the model has converged. The adopted value of tol allows considering cases in which the model is stable only over a short interval. Increasing the value of tol implies reducing the number of epochs required for convergence, but it decreases the final classification performance. The opposite occurs when we reduce the parameter.

Table 3. Communication requirements to train the models until convergence in different datasets and sample distribution settings considering the Total Data Transmitted (TDT).

		IID		**Non-IID 5 Classes**		**Non-IID 2 Classes**	
Dataset	Model	$\mathbb{E}[e]$ (#)	TDT (MB)	$\mathbb{E}[e]$ (#)	TDT (MB)	$\mathbb{E}[e]$ (#)	TDT (MB)
CIFAR-10	ATHENA-FL	7	37.635	12	64.517	14	75.270
	FedAVG (MobileNet)	30	378.106	23	289.881	8	100.828
	Economy (%)	-	**90.05**	-	**77.74**	-	**25.35**
MNIST	ATHENA-FL	4	21.506	24	129.034	4	21.506
	FedAVG (MobileNet)	7	88.225	58	731.005	65	819.230
	Economy (%)	-	**75.62**	-	**82.35**	-	**97.37**
FMNIST	ATHENA-FL	5	26.88	81	435.49	21	112.91
	FedAVG (MobileNet)	44	554.56	151	1903.10	80	1008.30
	Economy (%)	-	**95.15**	-	**77.12**	-	**88.8**



(a) CIFAR-10 dataset.

(b) MNIST dataset.

Figure 12. Time to compute a local update. This time is equivalent to 5 local epochs for a client.

The accuracy results from analysis over the training epochs presented in Section 5.2.1 according to the adopted convergence criterion, allows the estimation of the value $\mathbb{E}[e_{dec}]$ and $\mathbb{E}[e_{cmc}]$ for the datasets in each experimental setup. We show the results of ATHENA-FL compared with MobileNet in Table 3. The MobileNetV2 model needs hundreds of epochs to converge. For conciseness, we only show the results of MobileNet due to its faster convergence in tens of epochs in most scenarios.

After estimating the number of epochs necessary for the models to converge, it is possible to compare them in its scenario using properly the Equations 5 and 6. The results show how many bytes each neural network model needs to transmit during the training, and finally, we compare the economy of both approaches of computing $1 - \overline{B_{ova}/B_{cmc}}$.

In the IID setting, the detectors converge in a few epochs, reducing over 75% of bytes transmitted. A second highlight is the data used to train and test the models. ATHENA-FL converges faster than MobileNet in MNIST and FMNIST, which have grayscale images and, therefore, use a single channel to represent the pixels.

We observe that in all scenarios, ATHENA-FL reduces the total amount of bytes transmitted during the training execution. In the worst-case scenario, our approach saves at least 25% of the bytes transmitted. For the best case scenario, we can reduce by approximately 97% the communication requirements for training the model.

5.4 Computational Time

Finally, we evaluate the computational time among the local updates. The local update is equivalent to 5 local epochs since each client updates its model 5 times in our scenario before sending it to the aggregation server. To be concise, we show the results for the CIFAR-10 and MNIST datasets because samples in the FMNIST dataset have the same shape as samples in the MNIST dataset. Besides, to measure the time to a local update, only the number of training samples produces an impact. Clients have the same number of training samples in IID and non-IID distributions. The only difference is the data distribution. Therefore, we exhibit the time for the IID distribution, equivalent to the others. Also, the comparison considers the models of Table 3. Figure 12 exhibits the computational time results for both datasets using the two models.

Once the detectors consist of shallow neural networks, thus it takes less time to compute the local update. When using the ATHENA-FL, each client experiences a mean delay value near 8 seconds in CIFAR-10, while this value increases to 10 seconds in the MNIST dataset. Meanwhile, MobileNet introduces a higher delay, close to 13 seconds in CIFAR-10 and 15 seconds in MNIST. Despite the MNIST dataset having fewer features, it has more samples, which explains the increased delay compared to the CIFAR-10 dataset.

The results of this experiment show another advantage of our proposal. To compare the total training time, we use the values of the total number of epochs exhibited in Table 3. It is worth noting that the local updates for different detectors can be executed in parallel as long as the system has computational resources. Thus, even though it is necessary to train more models, we do not need to multiply the computational time by the number of models in the system. For instance, using the MobileNet in the non-IID scenario with the MNIST dataset takes 65 epochs to train the model will generate a delay of $65 \times 15 = 975$ seconds, or more than 16 minutes. On the other hand, for the same scenario, ATHENA-FL generates a delay of $4 \times 10 = 40$ seconds, which is less than a minute. Nonetheless, in the IID scenario with the CIFAR-10 dataset, our detectors need $14 \times 8 = 112$ seconds, while the MobileNet needs $8 \times 13 = 104$ seconds to be trained. Despite the MobileNet learning faster in this case, the total time is closer, leading to nearly 2 minutes for each approach to train the models. Hence, ATHENA-FL reduces the training time in most of the experiment scenarios.

6 Conclusion and Future Work

We presented ATHENA-FL, a federated learning system that clusters clients to reduce data heterogeneity during training. Furthermore, the system applies the one-*versus*-all model to create a generic classifier, which shares knowledge among clusters. The results show that communication during the training epochs is efficient, reducing between 25.35% and 97.37% total transmitted bytes compared to the FedAVG approach with the MobileNet neural network. The accuracy of the one-*versus*-all model depends on the data distribution scenario used during the training step, being up to 10.9% higher in the best case and presenting a better performance than the model trained with FedAVG in most of the evaluated scenarios. ATHENA-FL also outperforms traditional FL in non-iid scenarios closer to those encountered in real-life while reducing the required communication overhead by over 99%. ATHENA-FL also reduces the training time in most of the experiment scenarios especially due to the capability of training detectors' models in parallel. In future works, we will detail the proposal's security guarantees, providing a protocol tolerant to malicious participants' updates.

Declarations

Authors' Contributions

Lucas Airam Castro de Souza: Conceptualization, Validation, Investigation, Writing – original draft, Writing – review & editing Gustavo Franco Camilo: Validation, Investigation, Writing – original draft, Writing – review & editing

Matteo Sammarco: Methodology, Writing – original draft, Writing – review & editing,

Miguel Elias Mitre Campista: Methodology, Writing – original draft, Writing – review & editing, Supervision Luís Henrique Maciel Kosmalski Costa: Methodology, Writing – original draft, Writing – review & editing, Supervision

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

There is a GitHub repository associated with the paper: https://github.com/GTA-UFRJ/ATHENA-FL

References

Beutel, D. J. *et al.* (2020). Flower: A Friendly Federated Learning Research Framework. *arXiv preprint arXiv:2007.14390*. DOI: 10.48550/arXiv.2007.14390.

Blondel, V. D. *et al.* (2008). Fast Unfolding of Communities in Large Networks. *Journal of Statistical Mechanics: Theory and Experiment*, pages 1–12. DOI: 10.1088/1742-5468/2008/10/P10008.

Chu, D., Jaafar, W., and Yanikomeroglu, H. (2022). On the Design of Communication-Efficient Federated Learning for Health Monitoring. *IEEE GLOBECOM*, pages 1–6. DOI: 10.1109/GLOBECOM48099.2022.10001077.

de Souza, L. A. C., Camilo, G. F., Campista, M. E. M., and Costa, L. H. M. K. (2023a). Hierarchical clustering of nodes for accuracy increase in federated learning. Techinical Report, Electrical Engineering Program, COPPE/UFRJ. Available at:https://www.gta.ufrj.br/ftp/gta/TechReports/SCM23.pdf.

de Souza, L. A. C., Camilo, G. F., Rebello, G. A. F., Sammarco, M., Campista, M. E. M., and Costa, L. H. M. (2023b). ATHENA-FL: Evitando a Heterogeneidade Estatística através do Um-contra-Todos no Aprendizado Federado. In *Anais do VII Workshop de Computação Urbana*, pages 40–53. SBC. DOI: 10.5753/courb.2023.717.

de Souza, L. A. C. *et al.* (2020). DFedForest: Decentralized Federated Forest. In *IEEE International Conference on Blockchain*, pages 90–97. DOI: 10.1109/Blockchain50366.2020.00019.

Dennis, D. K., Li, T., and Smith, V. (2021). Heterogeneity for the Win: One-Shot Federated Clustering. *arXiv preprint arXiv:2103.00697*. DOI: 10.48550/arXiv.2103.00697.

Djenouri, Y., Michalak, T. P., and Lin, J. C.-W. (2023). Federated Deep Learning for Smart City Edge-based Applications. *Future Generation Computer Systems*, 147:350–359. DOI: 10.1016/j.future.2023.04.034.

Duan, M., Liu, D., Ji, X., Wu, Y., Liang, L., Chen, X., Tan, Y., and Ren, A. (2022). Flexible Clustered Federated Learning for Client-Level Data Distribution Shift. *IEEE Transactions on Parallel and Distributed Systems*, 33(11):2661–2674. DOI: 10.1109/TPDS.2021.3134263.

Ester, M., Kriegel, H.-P., Sander, J., Xu, X., *et al.* (1996). A Density-based Algorithm for Discovering Clusters in Large Spatial Databases with Noise. In *KDD*, pages 226–231. Available at:https://cdn.aaai.org/KDD/1996/KDD96-037.pdf?source=post_page---------------------------.

Fraboni, Y., Vidal, R., Kameni, L., and Lorenzi, M. (2021). Clustered Sampling: Low-Variance and Improved Representativity for Clients Selection in Federated Learning. In *International Conference on Machine Learning*, pages 3407–3416. PMLR. Available at:https://proceedings.mlr.press/v139/fraboni21a.html.

Fu, L. *et al.* (2022). Client Selection in Federated Learning: Principles, Challenges, and Opportunities. *arXiv preprint arXiv:2211.01549*, pages 1–8. DOI: 10.48550/arXiv.2211.01549.

Ghosh, A., Chung, J., Yin, D., and Ramchandran, K. (2020). An Efficient Framework for Clustered Federated Learning. *arXiv preprint arXiv:2006.04088*. DOI: 10.48550/arXiv.2006.04088.

Howard, A. G., Zhu, M., Chen, B., Kalenichenko, D., Wang, W., Weyand, T., Andreetto, M., and Adam, H. (2017). MobileNets: Efficient Convolutional Neural Networks for Mobile Vision Applications. *arXiv preprint arXiv:1704.04861*. DOI: 10.48550/arXiv.1704.04861.

Krizhevsky, A., Nair, V., and Hinton, G. (2014). The CIFAR-10 Dataset. *online: http://www. cs. toronto. edu/kriz/cifar. html*, 55(5). Available at:`https://www.cs.toronto.edu/~kriz/cifar.html`.

Lai, F., Zhu, X., Madhyastha, H. V., and Chowdhury, M. (2021). Oort: Efficient Federated Learning via Guided Participant Selection. In *USENIX OSDI*, pages 19–35. Available at:`https://www.usenix.org/conference/osdi21/presentation/lai`.

LeCun, Y., Cortes, C., and Burges, C. J. (2010). MNIST Handwritten Digit Database. http://yann.lecun.com/exdb/mnist/. Available at:`https://yann.lecun.com/exdb/mnist/`.

Li, D., Lai, J., Wang, R., Li, X., Vijayakumar, P., Gupta, B. B., and Alhalabi, W. (2023a). Ubiquitous Intelligent Federated Learning Privacy-preserving Scheme under Edge Computing. *Future Generation Computer Systems*, 144:205–218. DOI: 10.1016/j.future.2023.03.010.

Li, H., Cai, Z., Wang, J., Tang, J., Ding, W., Lin, C.-T., and Shi, Y. (2023b). FedTP: Federated Learning by Transformer Personalization. *IEEE Transactions on Neural Networks and Learning Systems*. DOI: 10.1109/TNNLS.2023.3269062.

Li, T., Hu, S., Beirami, A., and Smith, V. (2021). Ditto: Fair and Robust Federated Learning through Personalization. In *International Conference on Machine Learning*, pages 6357–6368. PMLR. Available at:`https://proceedings.mlr.press/v139/li21h`.

Li, T., Sahu, A. K., Zaheer, M., Sanjabi, M., Talwalkar, A., and Smith, V. (2020). Federated Optimization in Heterogeneous Networks. *Proceedings of Machine Learning and Systems*, 2:429–450. Available at:`https://proceedings.mlsys.org/paper_files/paper/2020/hash/1f5fe83998a09396ebe6477d9475ba0c-Abstract.html`.

Liu, B., Ding, M., Shaham, S., Rahayu, W., Farokhi, F., and Lin, Z. (2021). When Machine Learning Meets Privacy: A Survey and Outlook. *ACM Computing Surveys (CSUR)*, 54(2):1–36. DOI: 10.1145/3436755.

Liu, L., Zhang, J., Song, S., and Letaief, K. B. (2020). Client-Edge-Cloud Hierarchical Federated Learning. In *International Conference on Communications*, pages 1–6. DOI: 10.1109/ICC40277.2020.9148862.

Luo, B. *et al.* (2022). Tackling System and Statistical Heterogeneity for Federated Learning with Adaptive Client Sampling. In *IEEE INFOCOM*, pages 1739–1748. DOI: 10.1109/INFOCOM48880.2022.9796935.

Ma, X., Zhu, J., Lin, Z., Chen, S., and Qin, Y. (2022). A State-of-the-Art Survey on Solving Non-IID Data in Federated Learning. *Future Generation Computer Systems*, 135:244–258. DOI: 10.1016/j.future.2022.05.003.

McMahan, B. *et al.* (2017). Communication-efficient Learning of Deep Networks from Decentralized Data. *Artificial Intelligence and Statistics*, pages 1273–1282. Available at:`https://proceedings.mlr.press/v54/mcmahan17a?ref=https://githubhelp.com`.

Neto, H. N. C., Dusparic, I., Mattos, D. M., and Fernande, N. C. (2022). FedSA: Accelerating Intrusion Detection in Collaborative Environments with Federated Simulated Annealing. In *International Conference on Network Softwarization (NetSoft)*, pages 420–428. IEEE. DOI: 10.1109/NetSoft54395.2022.9844024.

Nishio, T. and Yonetani, R. (2019). Client Selection for Federated Learning with Heterogeneous Resources in Mobile Edge. In *International Conference on Communications*, pages 1–7. DOI: 10.1109/ICC.2019.8761315.

Ouyang, X. *et al.* (2021). ClusterFL: a Similarity-Aware Federated Learning System for Human Activity Recognition. In *Proceedings of the International Conference on Mobile Systems, Applications, and Services*, pages 54–66. DOI: 10.1145/3458864.3467681.

Qin, T., Cheng, G., Wei, Y., and Yao, Z. (2023). HierSFL: Client-Edge-Cloud Collaborative Traffic Classification Framework based on Hierarchical Federated Split Learning. *Future Generation Computer Systems*. DOI: 10.1016/j.future.2023.07.001.

Rai, S., Kumari, A., and Prasad, D. K. (2022). Client Selection in Federated Learning under Imperfections in Environment. *AI*, 3(1):124–145. DOI: 10.3390/ai3010008.

Sandler, M., Howard, A., Zhu, M., Zhmoginov, A., and Chen, L.-C. (2018). MobileNetV2: Inverted Residuals and Linear Bottlenecks. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 4510–4520. DOI: 10.1109/CVPR.2018.00474.

Sanguineti, M. (2021). Cats VS Dogs Convolutional Classifier. Towards Data Science. Available at:`https://towardsdatascience.com/cats-vs-dogs-convolutional-classifier-44ec04c8eb7a`.

Sattler, F., Müller, K.-R., and Samek, W. (2020). Clustered Federated Learning: Model-Agnostic Distributed Multitask Optimization under Privacy Constraints. *IEEE Transactions on Neural Networks and Learning Systems*. DOI: 10.1109/TNNLS.2020.3015958.

Singh, S., Rathore, S., Alfarraj, O., Tolba, A., and Yoon, B. (2022). A Framework for Privacy-preservation of IoT Healthcare Data using Federated Learning and Blockchain Technology. *Future Generation Computer Systems*, 129:380–388. DOI: 10.1016/j.future.2021.11.028.

Tan, A. Z., Yu, H., Cui, L., and Yang, Q. (2022). Towards Personalized Federated Learning. *IEEE Transactions on Neural Networks and Learning Systems*, pages 1–17. DOI: 10.1109/TNNLS.2022.3160699.

Tang, Z., Hu, Z., Shi, S., Cheung, Y.-m., Jin, Y., Ren, Z., and Chu, X. (2021). Data Resampling for Federated Learning with Non-IID Labels. In *FTL-IJCAI'21*. Available at:`https://federated-learning.org/fl-ijcai-2021/FTL-IJCAI21_paper_3.pdf`.

Wang, H. *et al.* (2020a). Optimizing Federated Learning on Non-IID Data with Reinforcement Learning. In *IEEE INFOCOM*, pages 1698–1707. DOI: 10.1109/INFOCOM41043.2020.9155494.

Wang, J. *et al.* (2020b). Tackling the Objective Inconsistency Problem in Heterogeneous Federated Optimization. *NeurIPS*, 33:7611–7623. Available at:`https://proceedings.neurips.cc/paper/2020/hash/564127c03caab942e503ee6f810f54fd-Abstract.html`.

Xiao, H., Rasul, K., and Vollgraf, R. (2017). Fashion-MNIST: A Novel Image Dataset for Benchmarking Machine Learning Algorithms. *arXiv preprint arXiv:1708.07747*. DOI: 10.48550/arXiv.1708.07747.

Yang, Q., Liu, Y., Chen, T., and Tong, Y. (2019). Federated Machine Learning: Concept and Applications. *Transactions on Intelligent Systems and Technology (TIST)*, 10(2):1–19. DOI: 10.1145/3298981.

Zeng, D., Hu, X., Liu, S., Yu, Y., Wang, Q., and Xu, Z. (2023). Stochastic Clustered Federated Learning. *arXiv preprint arXiv:2303.00897*. DOI: 10.48550/arXiv.2303.00897.

Zhao, Y., Li, M., Lai, L., Suda, N., Civin, D., *et al.* (2018). Federated Learning with Non-IID Data. *arXiv preprint arXiv:1806.00582*. DOI: 10.48550/arXiv.1806.00582.

Zhong, Z. *et al.* (2022). FLEE: A Hierarchical Federated Learning Framework for Distributed Deep Neural Network over Cloud, Edge and End Device. *ACM TIST*, pages 1–24. DOI: 10.1145/3514501.

Zhu, Y., Markos, C., Zhao, R., Zheng, Y., and James, J. (2021). FedOVA: One-vs-All Training Method for Federated Learning with Non-IID Data. In *IEEE IJCNN*, pages 1–7. DOI: 10.1109/IJCNN52387.2021.9533409.