COmmunity-Based Organized Littering

PRIN: PROGETTI DI RICERCA DI RILEVANTE INTERESSE NAZIONALE

Bando 2022 PNRR

Deliverable D6.1

A decentralized, trusted framework inspired by federated and peer-to-peer systems, with proper APIs/connectors for the proficient use of the solution

https://prin-cobol.github.io

Funded by the European Union
NextGenerationEU

Project Number	:	P20224K9EK
Project Title	:	COBOL: COmmunity-Based Organized Littering

Deliverable Number	:	D6.1
Title of Deliverable	:	A decentralized, trusted framework inspired by federated and peer-to-peer systems, with proper APIs/connectors for the proficient use of the solution
Nature of Deliverable	:	Report / Other
Dissemination level	:	Public
License	:	Creative Commons Attribution 3.0 License
Version	:	1.0
Contractual Delivery Date	:	M12
Contributing Objective/Task	:	O6/Tx.y
Editor(s)	:	Leonardo Mariani (UNIMIB)
Author(s)	:	Luciano Baresi (PoliMI), Simone Bianco (UniMIB), Amleto Di Salle (GSSI), Ludovico Iovino (GSSI), Daniela Micucci (UniMIB), Leonardo Mariani (UniMIB), Luciana Brasil Rebelo dos Santos (GSSI), Maria Teresa Rossi (UniMIB), Raimondo Schettini (UniMIB)

Glossary, acronyms & abbreviations

Item	Description
COBOL	COmmunity-Based Organized Littering
FL	Federated Learning
DNN	Deep Neural Network
CNN	Convolutional Neural Network

Table of Contents

1 Introduction

This document addresses the problem of conceiving a decentralized, trusted framework for the proficient use of the Cobol solution. The main contribution of the document is towards the adoption of Federated (machine) Learning (FL) to allow for the proper consumption of the garbage images provided by users.

In general, one can think of a solution in which users provide garbage-related pictures through their devices. Given the distributed nature of the context behind Cobol, a single centralized solution to analyze these data does not pay off. We need to envision a framework that can better adapt to the distributed nature of the problem and that also allows for flexibility and (future) extensibility. This is why Cobol has been working on the idea of adopting FL solutions to carry out the task, and compare them against more “standard”, centralized options.

The adoption of a distributed solution is also a means to respect the privacy of data generators, that is, of the users who provide garbage pictures and also to address possible communication bottlenecks. The adoption of users' devices as the nodes of the federation would give the best solution for privacy preservation, but the distribution would be excessive and each node (user) would probably not contribute enough artifacts (images) to allow for proper local training. The federation would become too wide and too distributed, and we may also have problems with proper training on mobile devices. The adoption of intermediary nodes is a compromise that can help address both issues. The pictures are sent “only” to the *trusted* nodes, which act as concentrators and can carry out proper “localized” training. Users' devices are relieved from energy-eager activities, and we do not need to make training work on diverse mobile architectures.

For the sake of standardization and also to get well-defined APIs almost free, we investigated the use of existing FL frameworks instead of developing a new hybrid solution. Some preliminary experiments suggested the use of Flower[1] as the basis for the foreseen solution. The framework is widely used, comes with a well-crafted architecture, is supported by a big community, and is actively maintained: these are all fundamental elements for the sustainability of Cobol.

Flower was also very helpful for carrying out the experiments, as it offered a simple, and easy to use and extend, sandbox to execute our experiments. Our main interest thus has been the study of different existing FL algorithms to select the nodes of the federation given the different requirements at hand. In general, the interest refers to the problem of managing heterogeneity in federated learning. Although it is clear that COBOL foresees a standard and static infrastructure, we wanted to study and compare the different existing solutions to have a complete landscape of the alternatives and be able to envision a preliminary decision tree that can help COBOL, but can also have wider applicability.

This is the main part of the document. We started with some simple solutions and a simple dataset, and continue with more advanced and recent solutions and more appropriate datasets. As mentioned above, we also wanted to study the differences between centralized and distributed (federated) solutions, and we did this exercise with the garbage dataset contributed by the project itself and pave the ground for realistic evaluations that can be used in the context of the project.

The rest of the document is organized as follows:

- Section 2 provides an overview of the FL process;
- Section 3 details the issue of heterogeneity management;

[1] https://flower.ai

- Section 4 surveys state-of-the-art FL algorithms;
- Section 5 reports on our experimental campaign;
- Section 6 presents a decision-support tree to select a FL solution;
- Section 7 concludes.

2 Federated Learning: Step-by-Step Overview

FL is a machine learning framework addressing, among others, the challenge of data privacy by allowing collaborative model training across multiple devices. This approach enables participants to construct a common (global) machine learning model without sharing data among themselves or with the server, leading to a data privacy preserving algorithm. Instead, their devices locally process the data, compute updates, and only send the model updates to a central server, ensuring that personal photos never leave the user's device. This privacy reserving approach is crucial in fostering user trust and increasing engagement within the community. When participants feel assured that their data remains private, they are more likely to participate by uploading photos of littered areas, enriching the data set, and improving the model performance.

McMahan et al. [1] introduced the concept of Federated Learning. It was developed to improve the performance of data-driven models under the constraints of data privacy laws, which cannot be preserved in typical centralized machine learning setups. The initial motivation was to enable smart edge devices, like mobile phones, to collaboratively learn a global prediction model while keeping all training data on the same device.

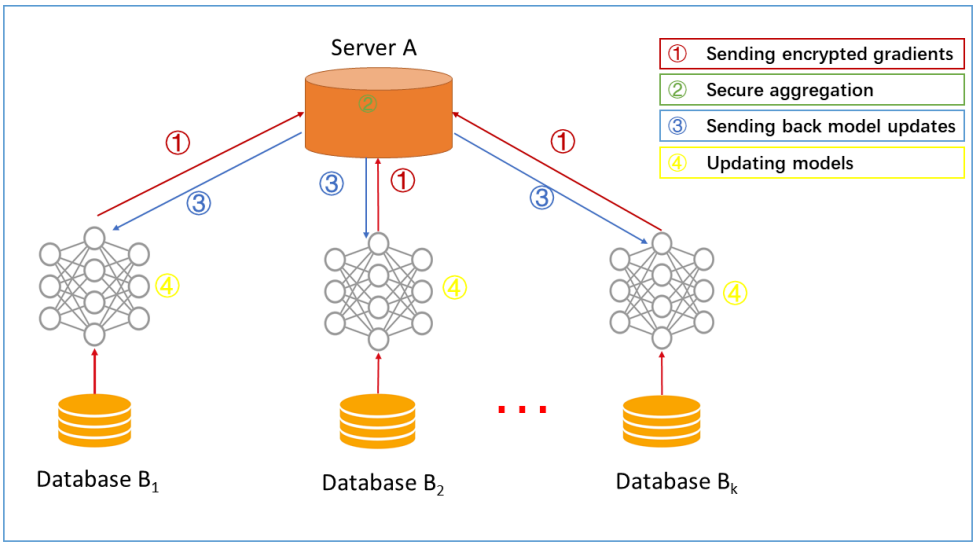


Figure 1. FL process overview.[2]

The steps of the baseline federated algorithm (FedAVG) as explained by [1] are shown in Figure 1 and summarised in the following:

1. **Sending models (or model updates):** the server shares an initial model (which can be empty) with a fraction of nodes (step 3 in Figure 1).
2. **Updating models:** a model is trained locally on multiple nodes (step 4 in Figure 1). Each node processes its own data locally to update the model's parameters.
3. **Sending encrypted gradients:** after local training, only the model updates (weights or gradients) are sent back to the server (step 1 in Figure 1).
4. **Secure aggregation:** the server aggregates these updates to improve the global model, which is then sent back to the nodes for further training.

The process of local training, model aggregation, and redistribution of the updated model continues across several rounds, enhancing the model's accuracy with each iteration.

3 Heterogeneity Management

FL introduces significant challenges due to the inherent heterogeneity of its decentralized architecture. This heterogeneity appears across various dimensions, which complicates the training and efficiency of federated models.

3.1 Types of Heterogeneity

In FL systems, we have many types of heterogeneity, Ye et al. [3] described some of these types:

- **Statistical Heterogeneity:** The data distribution across nodes is often Non-Independently and Identically (Non-IID), leading to divergent model updates and impacting overall model accuracy. This inconsistency can cause models to forget what they have already learned or not be able to learn some patterns (rare samples in a passive node). Addressing this is crucial for optimizing the performance and efficiency of federated learning systems.
- **Communication Heterogeneity:** Variability in network environments affects communication efficiency. Nodes with slower connections may significantly delay the aggregation process, sometimes leading to the discarding of model updates from these nodes. Employing strategies to manage this heterogeneity ensures the continuity and efficiency of the global model training, and ensures the participating of all the possible nodes.
- **Device Heterogeneity:** Differences in computational power and storage among devices can lead to uneven training times and updates. This often results in a training bottleneck, where slower devices delay the overall training process, highlighting the need for scalable and adaptive federated learning solutions.
- **Model Heterogeneity:** Nodes may use models of differing complexities to suit local computational capabilities or specific tasks, which will force flexible aggregation methods that can handle diverse model architectures. The aggregation process must be adaptive to the model's complexity and the nature of the data processed, ensuring that federated learning can be made more robust and scalable.

3.2 Managing Data Distribution Heterogeneity

The primary concern in FL, particularly from a data perspective, is the non-IID nature of data across different clients. Techniques to manage this include:

- **Node Selection Strategies:** Usually, the server chooses randomly a fraction of nodes to train the model locally at each round, some research shows that there are more effective ways to make this selection to overcome the problem of static heterogeneity.
- **Advanced Aggregation Algorithms:** Employing sophisticated aggregation methods that consider the amount and diversity of data each node contributes. This approach helps in creating a more representative and effective global model, enhancing the overall effectiveness of the learning process.
- **Data Augmentation and Synthetic Data Generation:** These techniques enhance the representativeness and balance of the training data across nodes, helping to mitigate the effects of skewed data distributions and ensuring that the benefits of federated learning are fully realized.

3.3 Addressing Model and Communication Heterogeneity

- **Model Heterogeneity:** To solve this problem, federated systems may incorporate meta-learning or multi-model training approaches that allow for personalized training while still contributing to collective knowledge. This enables each node to develop a model that is suitable to its specific data and computational context, thereby enhancing the adaptability and flexibility of the federated learning model.
- **Communication Heterogeneity:** Strategies such as asynchronous updates and differential synchronization can be employed to ensure that slower nodes do not unnecessarily delay the learning process. These methods allow nodes to contribute updates based on their individual communication capabilities, thus maintaining the continuity and efficiency of the global model training

4 State of the Art

Recent advancements in federated learning algorithms focus on optimizing communication and computation costs, managing heterogeneity, and enhancing model performance under various constraints. This section reviews several notable algorithms that represent the current state of the art in federated learning, detailing their approaches and contributions to the field.

4.1 Fast Federated Learning 'FFL' (2021)

FFL, or Fast Federated Learning [4], optimizes the federated learning process by dynamically balancing trade-offs between communication and computation and between communication and accuracy. It achieves this through innovative techniques like gradient compression and control over local updates, designed to speed up convergence while maintaining high accuracy.

A key contribution of FFL is its ability to minimize the upper bound of the error during each training round by jointly and dynamically adjusting the local iteration count and the gradient compression factor. The error bound for the k-th round is defined as follows:

$$\psi_k(\tau_k, s_k) = \frac{2[F(\mathbf{w}_k) - F_{\text{inf}}]}{\eta T_k} \left(Y_k + \frac{\alpha s_k}{\tau_k} \right) + \frac{\eta L \left(\frac{\sigma_1}{s_k} + \sigma_2 \right)}{M} + \eta^2 L^2 \left(\frac{\sigma_1}{s_k} + \sigma_2 \right) (\tau_k - 1), \quad (1)$$

Where:

- $\psi_k(\tau_k, s_k)$: The upper bound of the error in the k-th round.
- τ_k: The local update coefficient, which determines the number of local updates performed before communication.
- s_k: The sparsity budget for gradient compression, indicating the number of gradient components selected for transmission.
- $F(\mathbf{w}_k)$: The global loss function evaluated using the global model weights $\mathbf{w}_k$ in the k-th round.
- F_{inf}: The minimum value of the loss function, representing the convergence goal.
- η: The learning rate used during gradient updates.
- T_k: The total wall-clock time available for the k-th round.
- Y_k: The computational time required by the workers for local updates during the k-th round.
- α: A constant related to communication time per gradient component.
- L: The Lipschitz constant of the loss function, reflecting its smoothness.
- σ_1: The variance introduced by gradient compression.
- σ_2: The variance inherent to stochastic gradient descent (SGD).
- M: The number of participating workers in the federated learning system.

The learning process involves minimizing this upper bound, considering the constraints $\tau_k \in \{1, 2, \ldots, \tau_{\text{ub}}\}$ and $s_k \in [1, s_{\text{ub}}]$. Mathematically, this optimization can be expressed as:

$$\min_{\tau_k, s_k} \psi_k(\tau_k, s_k) \quad \text{subject to } \tau_k \in \{1, 2, \ldots, \tau_{\text{ub}}\},\ s_k \in [1, s_{\text{ub}}]. \quad (2)$$

FFL particularly works best in environments where communication costs are a primary concern, making it highly suitable for large-scale privacy-preserving machine learning applications such as those involving IoT devices. This method addresses the critical challenges of high communication overhead by efficiently managing gradient transmissions and local updates, demonstrating superior performance in reducing data transfer costs compared to traditional methods.

4.2 Progressive Federated Learning 'ProgFed' algorithm (2022)

ProgFed [5] introduces a progressive training framework for federated learning which reduces both computational and communication costs by incrementally increasing model complexity during training. This innovative approach exploits the natural stabilization of neural networks, starting with simpler, shallower submodels and gradually expanding to full-complexity models. This progressive growth allows significant savings in computational resources and communication overhead while maintaining high model performance.

ProgFed divides the model into multiple overlapping partitions, with each partition corresponding to a stage of training. In each stage, a submodel, denoted as Ms, is trained independently with local supervision. As training progresses, these submodels are expanded into more complex models by incorporating additional layers or blocks. The training process includes the following stages:

- **Stage 1: Shallow Submodel Training**
  The initial submodel $M1$, consisting of the simplest blocks ($E1$) and a lightweight head ($G1$), is trained. This significantly reduces computation and communication costs, as only a fraction of the total model parameters are involved.
- **Stage 2 to S: Progressive Growing**
  At each subsequent stage sss, the current submodel Ms incorporates additional blocks (Es) and an updated head (Gs). The weights of previously trained blocks are reused, ensuring continuity and efficient knowledge transfer.
- **Final Stage: Full Model Training**
  Once the network reaches its full complexity (Ms), end-to-end training is performed to fine-tune all layers.

This approach is compatible with various federated learning optimizations, such as FedAvg [1] and FedProx [6], and can be extended to both feedforward architectures and complex models such as U-nets.

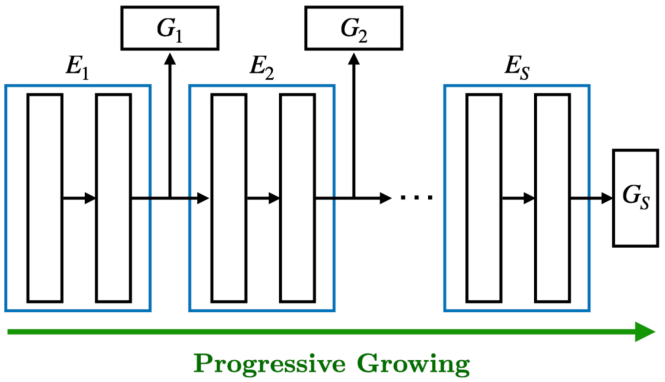


Figure 2. *Progressive Growing in Feed-forward Networks: the model expands stage by stage until the full model is achieved.*

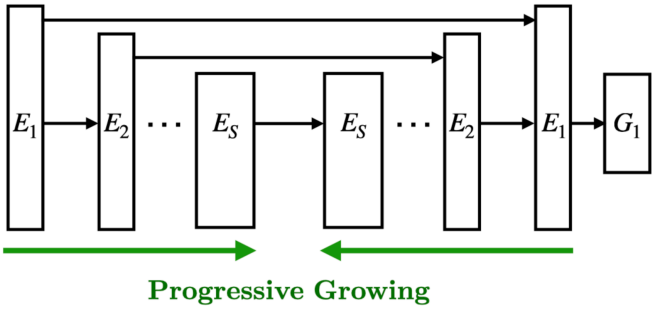


Figure 3. *Symmetric Progressive Growing in U-nets: Both encoder and decoder are grown outward until the complete U-net model is trained.*

Feedforward Networks

In feedforward networks, the network is progressively grown by training each submodel with localized supervision provided by temporary heads. These temporary heads are discarded once their corresponding stages are completed, and subsequent stages build upon the pre-trained layers.

U-nets

For U-nets, ProgFed adopts a symmetric growing strategy, where both the encoder and decoder are progressively expanded. Intermediate supervision is applied at each stage to effectively guide the training process. This strategy is particularly beneficial for segmentation tasks, where U-nets have demonstrated exceptional performance.

Results and Impact

ProgFed achieves significant savings in computational and communication costs without sacrificing performance. Key benefits include:

- Up to **25% reduction in computation costs** during classification tasks.
- Up to **63% savings in communication costs** in segmentation tasks.
- Compatibility with existing compression techniques and federated optimizations.

This innovative framework demonstrates its suitability for resource-constrained federated learning scenarios, such as IoT devices and medical imaging applications. Using the principles of progressive learning, ProgFed offers a flexible and efficient alternative to traditional federated learning approaches.

4.3 Cache-Enabled Federated Learning (CacheFL, 2023)

Liu et al. [7] introduced CacheFL which is a novel caching mechanism designed to enhance the efficiency of federated learning systems by reducing the total wall-clock training time, a critical metric in FL. Instead of requiring all clients to download the latest global model from the central server in every iteration, CacheFL allows a subset of clients to utilize cached, slightly old global models stored locally at edge devices or access points. This design significantly reduces communication overhead, particularly for clients with limited bandwidth or high communication latency.

The main innovation of CacheFL lies in its flexible caching strategy, which partitions clients into two groups: cache-enabled clients (K-cache), which use cached global models, and server-dependent clients (K-server), which rely on the latest global model. This partitioning mitigates the straggler effect, where slow clients delay the entire training process. By enabling cache-enabled clients to begin their local updates earlier with cached models, the per-iteration delay is reduced, thereby optimizing training efficiency.

Algorithm 1: CacheFL (cache-enabled FedAvg)

Input: Initial model $\mathbf{w}^0$, learning rate η, batch size B

for $t \in \{0, 1, \ldots, T-1\}$ **do**
  for *Client* $k \in \mathcal{K}$ **do**
    Let $\mathbf{w}_k^{(t,0)} = \mathbf{w}^t$, if $k \in \mathcal{K}_{\text{server}}$
    Let $\mathbf{w}_k^{(t,0)} = \mathbf{w}^{t-1}$, if $k \in \mathcal{K}_{\text{cache}}$
    for $\tau = 0, \ldots, \tau_{max} - 1$ **do**
      SG: $\mathbf{g}_k^{(t,\tau)} = \frac{1}{B} \sum_{i=1}^{B} \nabla f(\mathbf{w}_k^{(t,\tau)}; \mathbf{u}_i, v_i)$
      Local update: $\mathbf{w}_k^{(t,\tau+1)} = \mathbf{w}_k^{(t,\tau)} - \eta \mathbf{g}_k^{(t,\tau)}$
    end
  end
  for *Server after receiving* $\mathbf{w}_k^{(t,\tau_{max})}$ *for all* $k \in \mathcal{K}$ **do**
    Update global model $\mathbf{w}^{t+1} = \sum_{k \in \mathcal{K}} d_k \mathbf{w}_k^{(t,\tau_{\max})}$
  end
end
return $\mathbf{w}^T$

The CacheFL training process extends the traditional FedAvg algorithm by incorporating cache-enabled clients. As described in Algorithm 1, the initialization of the model for each client is determined by their group: server-dependent clients initialize with the latest global model, while cache-enabled clients use the previously cached global model. Local updates are performed using stochastic gradient descent (SGD), and the global model is updated on the server by aggregating the locally updated models.

Key characteristics of CacheFL include:

• **Caching Strategy:** The global model is stored in caches located at clients, access points, or the server, depending on the network topology. Cached models are updated at regular intervals, ensuring the upper limit of old models usage.

• **Trade-Off Management:** While caching introduces some staleness in global models, the slight increase in the number of iterations is offset by the significant reduction in per-iteration delay. CacheFL carefully balances this trade-off to minimize the total training time.

• **Applicability:** CacheFL is especially effective in scenarios with resource-constrained networks and heterogeneous client capabilities, as it reduces communication and computational overhead while maintaining convergence guarantees.

Experimental results demonstrate that CacheFL achieves up to a 28% reduction in the per iteration delay compared to traditional FedAvg while maintaining competitive accuracy. This makes CacheFL particularly suitable for federated learning scenarios with constrained bandwidth and heterogeneous client environments.

4.4 FedMDC: Multicenter Federated Learning with Model Decoupling

B. Wang et al. [8] address the challenges of data heterogeneity in federated learning by implementing a model decoupling framework designed to enable personalized model training while leveraging the advantages of a global model. FedMDC (the name of the new algorithm) enhances the handling of non-IID (non-independent and identically distributed) data across distributed clients by introducing a client clustering mechanism based on a low-dimensional representation of their data. By grouping clients into clusters, FedMDC creates local subglobal models, enabling better alignment with the statistical properties of the data at each client.

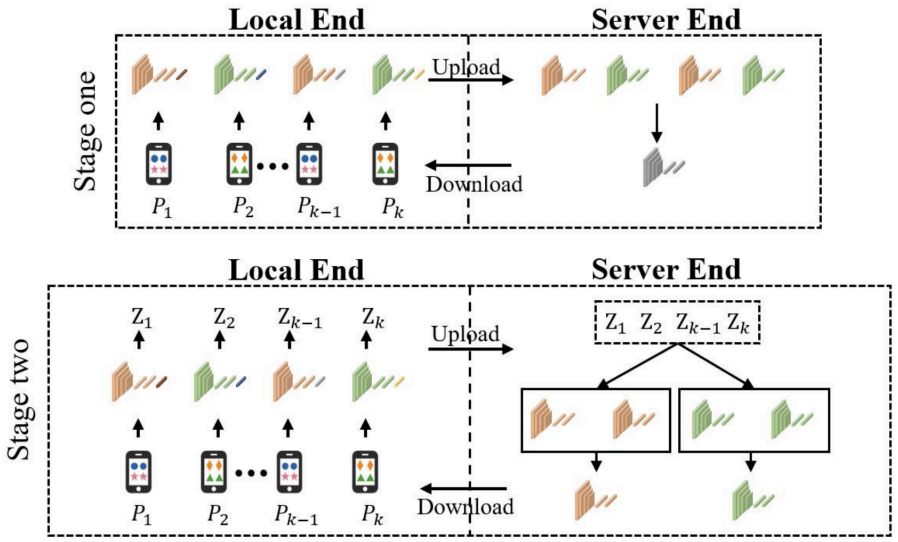


Figure 4. The FedMDC framework: Stage one involves local feature extraction and aggregation, while stage two clusters clients based on data similarity and trains sub-global models.

Framework Overview: The FedMDC framework operates in two key stages, as illustrated in Figure 4:

• **Stage One:** Local End Pre-training. Clients locally pre-train feature extractors P_1, P_2, ... , P_k using their private datasets. These pre-trained feature representations are uploaded to the server, where a global feature extractor is aggregated. This initial step reduces the heterogeneity among clients while preparing their local models for clustering.

• **Stage Two:** Client Clustering and Submodel Training. Based on the uploaded feature representations, the server clusters clients into groups with similar statistical data properties. Each cluster trains a subglobal model tailored to its members. The server aggregates these subglobal models into a unified global model by combining intermediate representations (Z_1, Z_2, ... , Z_k). The resulting models are redistributed to clients for further refinement. Handling Non-IID Data: The clustering mechanism in FedMDC significantly mitigates the challenges posed by non-IID data. By grouping clients with similar data distributions, the algorithm ensures that local updates contribute more effectively to subglobal models. This decoupled training strategy allows for the creation of personalized models that maximize the utility of local data while maintaining compatibility with the global model.

Communication Overhead: FedMDC introduces additional communication overhead due to the need for feature aggregation and client clustering. While this may increase communication costs compared to traditional methods, the clustering process and the use of

sub-global models offset these costs by significantly improving model convergence and reducing the overall training time.

Applications and Benefits: FedMDC is particularly effective in environments requiring personalized models, such as healthcare and IoT applications, where the data distributions of the clients are highly variable. Experimental results demonstrate that FedMDC improves the performance of local models by aligning them more closely with their respective data distributions, achieving superior accuracy compared to baseline federated learning methods.

4.5 FedGKD: Federated Global Knowledge Distillation

FedGKD, or Federated Global Knowledge Distillation, addresses the challenge of client drift in heterogeneous federated learning environments through a novel global knowledge distillation approach [9]. By leveraging historical global models as teachers, FedGKD guides the training of local models to better align with the global objective, thereby mitigating the divergence caused by non-IID data distributions across clients. This innovative method improves the convergence rate of the global model and maintains high accuracy and robustness in federated settings characterized by significant data heterogeneity.

FedGKD employs an ensemble-based knowledge distillation mechanism. Historical global models stored in the server buffer are aggregated to form an ensemble global model, which serves as a guide for local training. In each communication round, the ensemble model is broadcast to the participating clients. Clients utilize this ensemble model to refine their local training through knowledge distillation, minimizing the discrepancy between the output of the global ensemble model and the local models. This process ensures that local models capture diverse features while adhering to the global learning objective.

Figure 5 provides an overview of the FedGKD system. The architecture consists of two main stages:

1. Global Knowledge Ensemble: The server maintains a buffer of historical global models. These models are aggregated through parameter averaging to form the global ensemble model. This aggregated model encapsulates a more comprehensive representation of global knowledge.

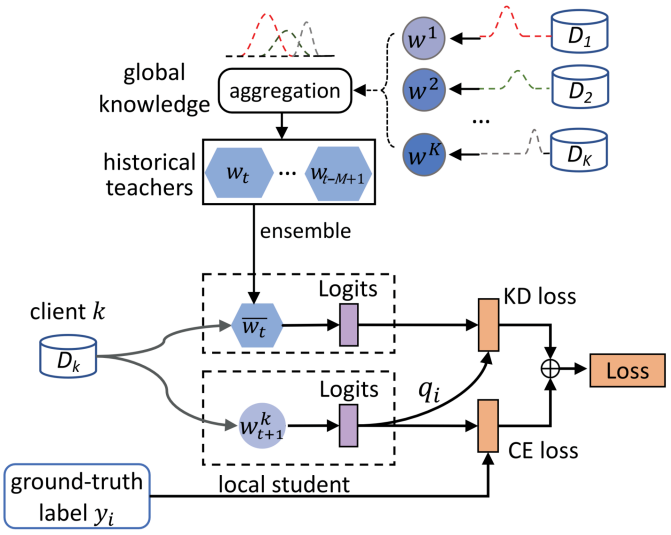


Figure 5. Overview of the FedGKD framework. The historical global models stored in the server buffer are aggregated into an ensemble model, which is broadcast to clients for local training. The clientsdistill the global knowledge into their local models through a combination of CE and KL losses.

2. Knowledge Distillation to Local Models: The ensemble global model is distributed to the participating clients. On the client side, the distillation process involves minimizing a combination of cross-entropy (CE) loss and Kullback-Leibler (KL) divergence loss, as shown in the equations:

$$\text{Objective Function:} \quad \min_{w} \quad F_k(w) + \frac{\gamma}{2n_k} \sum_{i=1}^{n_k} KL(h_k(w_t, x_{ki}) | 0 h_k(w, x_{ki})), \tag{3}$$

where w represents the local model, w_t is the global model in round t, and h_k denotes the logits of the model.

This two-stage process addresses the client-drift problem by ensuring that local models align closely with global knowledge while maintaining flexibility for local adaptations. FedGKD achieves these improvements without the need for additional privacy compromises or substantial changes in the model architecture. Experimental evaluations on various datasets demonstrate that FedGKD outperforms state-of-the-art baselines in terms of both accuracy and convergence speed, particularly in scenarios with high data heterogeneity. This makes it a versatile and effective solution for practical federated learning deployments.

4.6 Low Node Selection in Federated (LCNSFL, 2024)

LCNSFL (Low Node Selection in Federated Learning) addresses the challenges of non-IID data distributions and resource constraints in smart cities through a novel proximity-based node selection strategy within the Space-Air-Ground Information Network (SAGIN). This approach specifically targets environments characterized by diverse IoT devices with varying communication and computational resources, with the aim of reducing federated training time and energy costs while improving global model convergence and accuracy.

The proposed framework leverages near-edge optimization to prioritize IoT devices closer to the network edge for participation in federated training. This strategy reduces latency and transmission costs by decreasing the reliance on devices with weaker connections to the central server. By carefully selecting nodes based on their proximity and computational capacity, LCNSFL ensures efficient resource utilization and robust performance of the global model.

The workflow of LCNSFL, depicted in Figure 6, begins with the initialization of device information and the downloading of the global model to participating devices. Subsequently, the server collects information about the status of the system, such as the operational state of the device and the network conditions, which are input into an actor network. The actor network evaluates and ranks devices based on their action probabilities, selecting the top-K devices for participation in the current training round.Top-K Selection: In LCNSFL, the top-K selection mechanism identifies the most suitable K devices from a larger pool of available nodes for participation in a given training round. These devices are ranked on the basis of their action probabilities. The ranking criteria include factors such as proximity to the network edge, computational power, battery level, and quality of the connection. By selecting the top K devices, LCNSFL ensures efficient utilization of resources, reduces latency, and improves the overall effectiveness of the federated learning process. These selected devices perform local training, after which the server aggregates the updated models to refine the global

model. This iterative process continues until the global model achieves the target accuracy, optimizing training efficiency while maintaining high accuracy in heterogeneous settings.

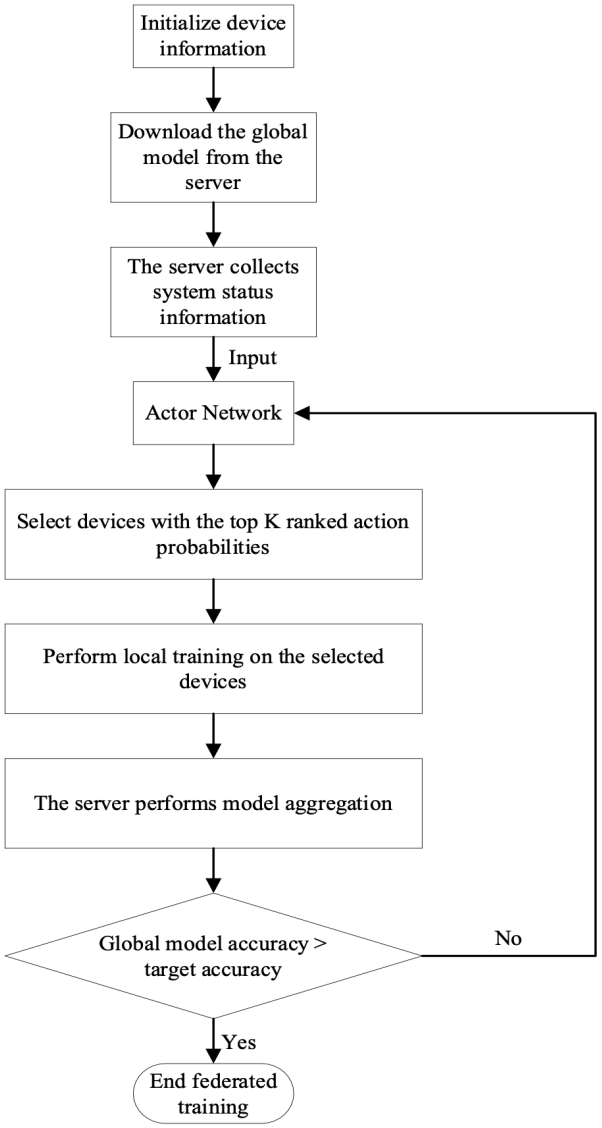


Figure 6. Workflow of LCNSFL for federated learning node selection based on near-edge optimization.

The experimental results, as described in [10], demonstrate that LCNSFL performs better compared to traditional methods such as random selection and FedProx. Specifically, LCNSFL significantly reduces communication rounds, energy costs, and overall training time in scenarios with varying degrees of non-IID data distributions. This efficiency makes it an ideal solution for federated learning in large-scale networks, particularly in smart cities where communication and computational resources are limited.

4.7 Federated Learning with Adaptive Weighted Model Aggregation (FLAMA, 2023)

FLAMA was introduced in [11] and is a dynamic approach to address the challenges of data heterogeneity and model convergence in federated learning systems. Unlike traditional aggregation strategies, FLAMA adapts model aggregation weights in each training round based on the usefulness of the data samples provided by each node. This mechanism

ensures that nodes contributing more relevant data—such as recent user interactions in recommendation systems are prioritized, thereby improving the global model's accuracy and convergence efficiency.

Key Components of FLAMA: As illustrated in Figure 7, the FLAMA framework operates in two main stages:

1. Useful Data Rate (UDR) Reporting: Each client calculates its Useful Data Rate (UDR), which represents the proportion of data samples considered relevant for training in the current round. This UDR value, along with the model updates, is sent to the server.

2. Adaptive Weight Calculation: The server uses the UDR values to calculate aggregation weights for each client. Clients with higher UDR values are assigned larger aggregation weights, ensuring that their contributions are emphasized in the global model aggregation.

The server also performs a global model accuracy test and adjusts the minimum aggregation weight (minAW) to balance the representation of useful and normal data.

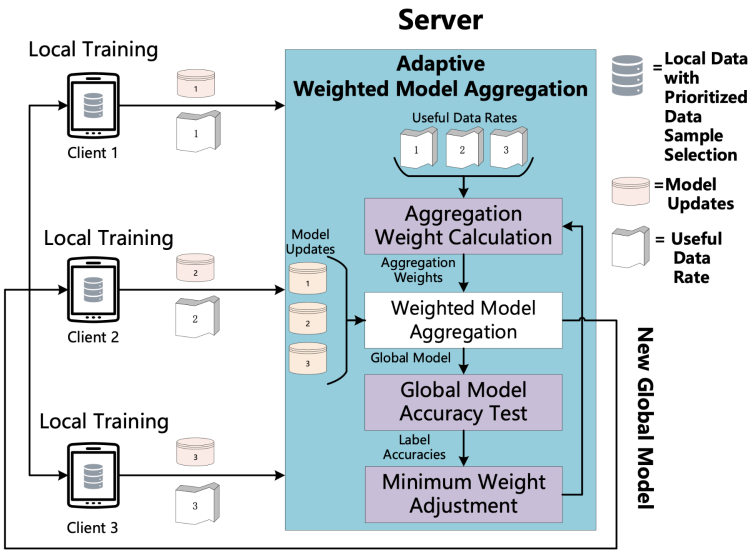


Figure 7. The FLAMA framework: Adaptive weighted model aggregation based on useful data rates (UDR) to prioritize informative client contributions.

Aggregation Weight Adjustment: FLAMA employs a dynamic adjustment mechanism for the minimum aggregation weight (minAW). If the global model shows bias toward either useful or normal data, the server modifies minAW to correct this imbalance. This ensures that both special and normal clients are adequately represented in the training process, thereby maintaining fairness and optimizing model performance.

Performance Benefits: Experimental results demonstrate that FLAMA achieves a significant improvement in model accuracy compared to FedAvg and other fixed-weight aggregation methods. By prioritizing useful data through the UDR metric, FLAMA reduces the number of training rounds needed for convergence while maintaining robust performance across diverse client data distributions. FLAMA's dynamic adaptation of aggregation weights ensures that federated learning systems can efficiently handle data heterogeneity while achieving rapid convergence and high model accuracy, making it a valuable solution for modern federated learning challenges

5 Empirical Evaluation

Experiments carried out so far within the project assess the effectiveness of various FL strategies, particularly focusing on their capacity to manage data heterogeneity and optimize computational and communication resources. This experimental campaign provides insights into choosing the most efficient FL algorithm for our training process, since we expect significant heterogeneity among the pictures taken by the users and model training must still be completed as quickly as possible.

Our approach involved a comprehensive analysis of node selection and workload optimization techniques within a FL framework, employing the Flower framework for implementation and testing.

The experimental campaign is structured into three phases involving the training of increasingly complex classification models against increasingly challenging datasets, as summarised in the following:

- **Phase 1**: Multilayer Perceptron (MLP) model, MNIST dataset.
- **Phase 2**: Convolutional Neural Network (CNN), cifar-10 dataset.
- **Phase 3**: ResNet18 model, TACO dataset.

5.1 Phase 1: Design of Experiments

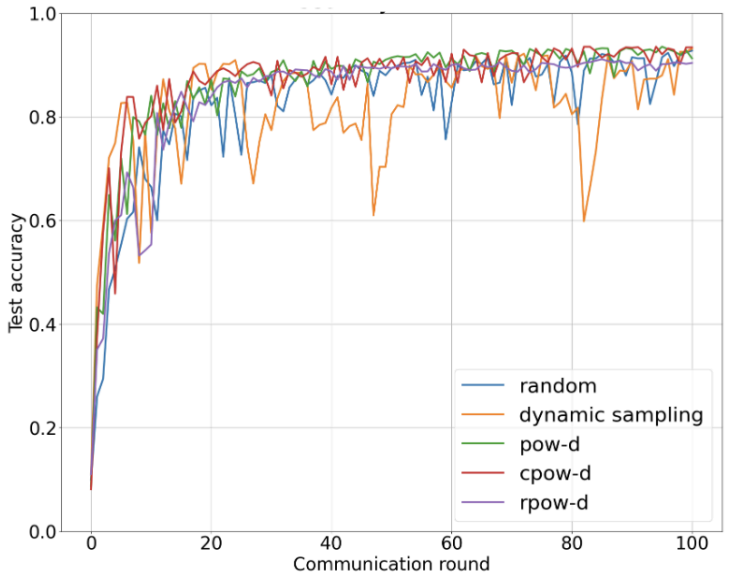
Our experimental setup utilized a multilayer perceptron (MLP) model trained on the MNIST dataset, which is standard for benchmarking FL algorithms due to its simplicity and suitability for illustrating the challenges of non-IID data distribution. We implemented four node selection algorithms: Dynamic Sampling, pow-d, cpow-d, and rpow-d, alongside the FedAvg algorithm as a baseline for comparison. For workload optimization, we explored four techniques: Static Optimizer, Uniform Optimizer, Round Time Optimizer, and Equal Computation Time Optimizer.

Each node in our federated network was simulated to have varying computational capabilities and data availability, reflecting a realistic environment where devices such as smartphones and tablets contribute to the learning process. The diversity in node capability and data distribution allowed us to test the pros and cons of each strategy under different levels of system and statistical heterogeneity.

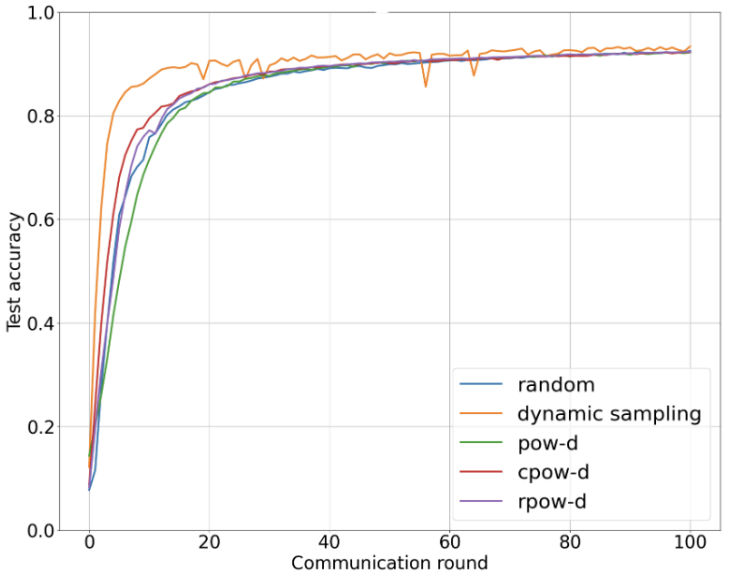
5.2 Phase 1: Results

Our findings revealed that dynamic node selection strategies improved the convergence speed and model accuracy over the baseline FedAvg approach. These strategies effectively addressed the non-IID nature of data distribution by prioritizing nodes that would most benefit global model updates at each training round.

Workload optimization strategies showed varying degrees of success in managing computational resources efficiently. The Round Time Optimizer emerged as particularly effective, optimizing the allocation of computational tasks based on the capabilities of each node, thereby reducing bottlenecks and improving overall system efficiency.



(a) $\alpha = 0.5$.



(b) $\alpha = 100$.

Figure 8. Accuracy evolution over training rounds (for both alpha= 0.5 and 100).

However, the numerical result from the simulations showed that the performance of a specific algorithm compared to others can be different depending on the level of heterogeneity of the data distribution as shown in Figure 8 which shows that dynamic sampling reached faster convergence of the model when the data are more distributed in iid, but was slower when the data are not distributed in iid.

While for the workload optimizers there was a type of trade between the highest accuracy achieved and the required training time, the training time is a very important metric in scenarios where the nodes are not stables in terms of electrical power or connection links. Table 1 shows the numerical results of the different types of optimizers, in terms of accuracy and max training time and the variance of the training time.

α	Strat.	M. Acc	F. Acc	Tr. Loss	T_{max}	$\mathcal{S}^2$
4*.05	*static*	**0.9240**	**0.8932**	**0.3447**	34.57	171.21
	uniform	0.8932	0.8921	0.3978	5.98	4.53
	RT	0.9099	0.8338	0.4968	16.72	35.76
	ECT	0.8641	0.8526	0.5310	**1.85**	**0.44**
4*1	*static*	**0.9300**	**0.9215**	**0.2693**	28.08	75.76
	uniform	0.8902	0.8711	0.4336	7.51	6.52
	RT	0.9121	0.8881	0.3632	11.29	12.16
	ECT	0.8787	0.8766	0.4743	**1.85**	**0.30**
4*10	*static*	**0.9257**	**0.9247**	**0.2650**	13.51	8.93
	uniform	0.8712	0.8712	0.5023	4.22	1.61
	RT	0.9057	0.9049	0.3292	6.14	1.60
	ECT	0.8697	0.8697	0.4963	**1.85**	**0.01**
4*100	*static*	**0.9222**	**0.9212**	**0.2723**	12.65	6.82
	uniform	0.8549	0.8549	0.5519	3.55	1.02
	RT	0.9053	0.9053	0.3309	5.37	0.64
	ECT	0.8762	0.8762	0.4634	**1.87**	**0.01**

Table 1. Workload optimization strategies (best results in bold).

5.3 Phase 2: Design of Experiments

Building on the initial findings from Phase 1[12], we expanded our experiments to include a more complex model and a different dataset to further assess the robustness and scalability of our federated learning strategies under various conditions. Feedback from peer reviews highlighted the need to explore additional models and datasets to deepen our understanding of the effectiveness of these strategies which also align with the nature of COBOL project requiring a deep neural network model.

To this end, we employed a Convolutional Neural Network (CNN), designed to handle complex image data more effectively. This CNN consists of multiple convolutional and pooling layers, interspersed with dropout layers to combat overfitting. We trained this model using the cifar-10 dataset, which presents more complexity than MNIST dataset used in Phase 1, providing a challenging environment for evaluating our FL approaches.

The CNN architecture was defined as follows:

• Three convolutional layers, each accompanied by varying rates of dropout.

• A final stage of global average pooling followed by dense layers, tasked with classifying images into ten distinct categories.

A configurable Latent Dirichlet Allocation (LDA) distribution based on parameter DirK(α) can be used to construct heterogeneous data partitions among nodes, where parameter α controls the degree of data heterogeneity. If $\alpha \rightarrow \infty$, all clients have identical distribution. If $\alpha \rightarrow 0$, each client holds samples from only one class [12]. We conducted the experiments across a spectrum of heterogeneity levels:

• Alpha = 0.5 and Alpha = 1, corresponding to scenarios with high heterogeneity.

• Alpha = 10 and Alpha = 100, corresponding to lower heterogeneity.

5.4 Phase 2: Results

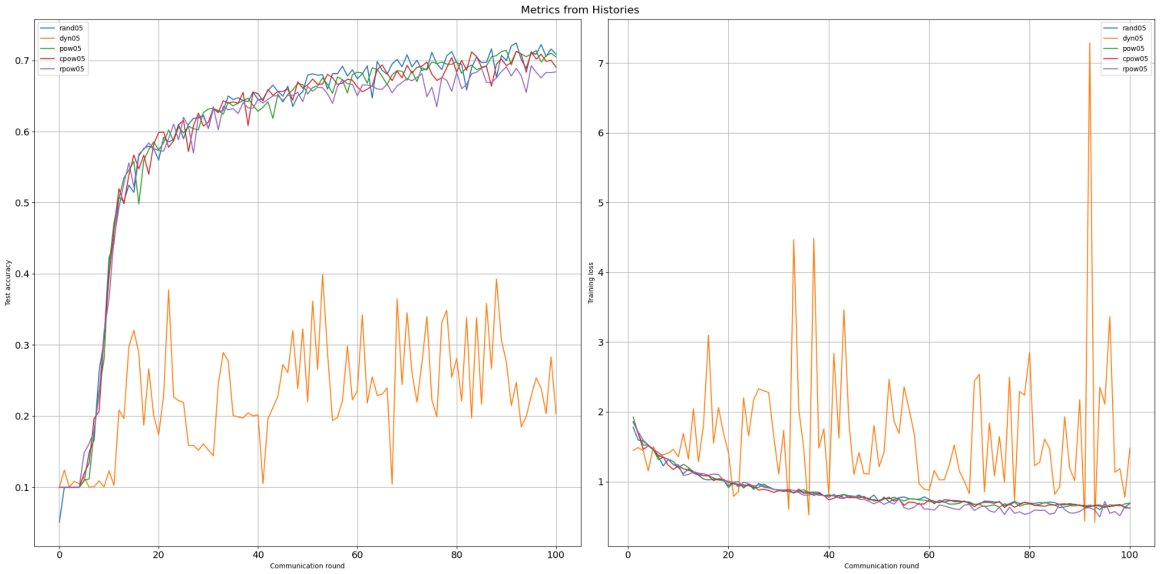


Figure 9. Test accuracy comparison when alpha = 0.5.

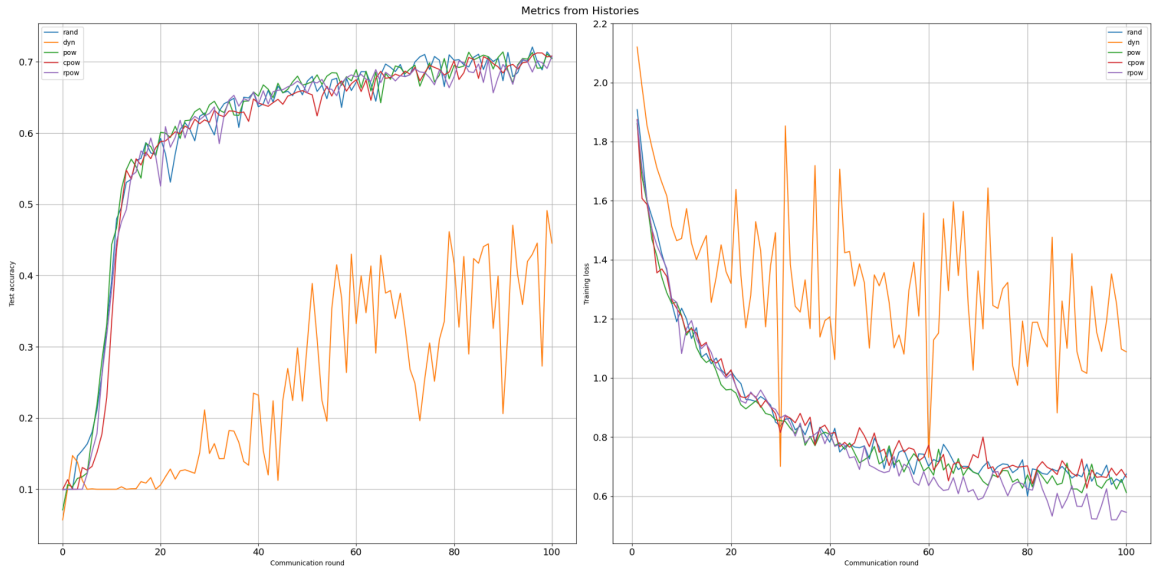


Figure 10. Test accuracy comparison when alpha = 100.

Our findings indicated that while the dynamic sampling strategy achieved faster convergence comparing centralized test accuracies with other algorithms, in simple model training as shown in [12], it did not have the same result with the complex CNN model as shown in Figures 9 and 10 which show the test accuracy evolution over the training rounds, and the training loss. The pow-d strategy and its variants, which prioritize the nodes expected to offer the most useful information, showed superior performance at all levels of heterogeneity.

The diverse results obtained from the CIFAR-10 dataset with a complex CNN model compared to Phase 1 results showed the necessity of an advisory system to guide the selection of the most suitable federated learning algorithm based on specific conditions for each situation. Performance variations highlight the influence of multiple factors, including task complexity, network size, and data distribution among nodes. This variability emphasizes the importance of tailored algorithm selection to optimize performance in different federated learning scenarios.

5.5 Phase 3: Design of Experiments

The purpose of Phase 3 is to to train a complex deep neural network (ResNet18) to classify images containing litter and clean background. These experiments were done to extend the results obtained by [1] and prove that FL achieves similar accuracy results compared to the centralized training setting with a deep model and a more complex classification problem.

The dataset is a set of image crops taken from TACO dataset classified as litter or background. The crops are already resized to 224x224 which is a common input size for many CNN architectures (ResNet18 included). The dataset was divided into seven distinct parts, each representing a different node in our federated learning model. This division was intentionally designed to simulate the potential heterogeneity among nodes, with each part containing different types of background and different numbers of data samples.

We initiated training the ResNet18 model from scratch in a federated setting. The model was trained for 100 rounds of training, mirroring the real-world scenario where multiple nodes contribute to the learning process without sharing their local data. For comparison, we also trained the ResNet18 model using all the data collected in a centralized manner.

Both sets of experiments (federated and centralized) were done under identical training conditions to maintain consistency and fairness in evaluation:

• Number of Training (Rounds, Epochs): 100

• Batch Size: 32

• Learning Rate: 0.001

Due to computational limitations, the FL simulation involved selecting 5 out of the 7 nodes in each training round (using the baseline algorithm 'FedAVG'). This approach simulates a real life situation were the node's connection link might drop down and highlights the ability of FL to converge even when the environment is not perfect.

5.6 Phase 3: Results

The outcomes of these experiments were similar in terms of test accuracy, demonstrating that the FL model could achieve nearly equivalent performance to the centralized model, despite the inherent challenges of partial node participation and data heterogeneity.

Figure 11 shows the performance of FL compared to traditional centralized training, also in the presence of heterogeneity and with some nodes' connections dropping during training.

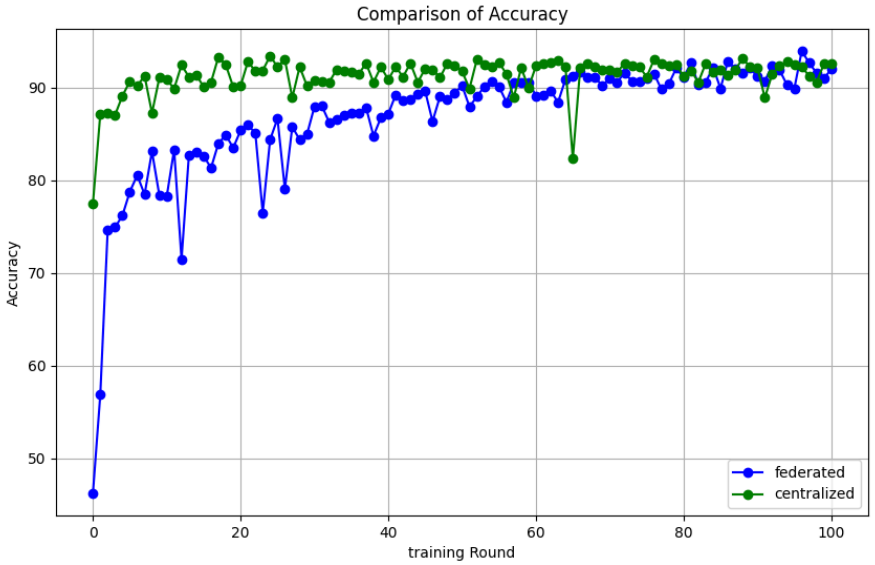


Figure 11. Test accuracy comparison between centralized and federated settings.

6 Selecting the Right Federated Learning Approach

Selecting the appropriate FL algorithm based on the specific requirements and constraints of the environment is crucial. To support this process, we created a structured decision-support system (specifically, a decision tree) that outlines the key trade-offs and performance characteristics of different algorithms. The decision tree guides stakeholders toward informed decisions when deploying FL systems in diverse and potentially resource-constrained environments. By highlighting the core attributes of each algorithm, the decision tree identifies the most suitable approach based on key questions regarding potential application-specific concerns.

As an initial step toward the decision tree, we collate state-of-the-art FL algorithms and summarise their strengths and potential weaknesses. This analysis lays the foundation for a more detailed and comprehensive study.

As a starting point, we evaluate three key factors: communication efficiency, computational load, and the ability to handle data heterogeneity. By categorizing and comparing these algorithms, we assess the trade-offs of different FL deployments.

algorithm	communication	computation	Non-iid (heterogeneity)	code	In general
FFL(2021)	Optimizes communication via gradient compression and controlling local updates. It is ideal for reducing data transfer.	Controls computational load by adjusting local updates, effectively managing device capabilities to make a balance between comm and comp.	Not specified		Highly suitable for environments where communication costs are critical.
ProgFed (2022)	Reduces initial communication costs by progressively increasing model complexity.	Improves computational efficiency over time, distributing the load more evenly.	Not specified		Perform the best in conditions where the communication and computational costs are critical, and the used model is large for node training.
CachFL (2023)	Reduces training communication by caching old models.	Not related to comp, but it required more memory in the edges to save the old model.	Not specified		Proved to reduce the communication cost.
FedMDC (2024)	Increased communication because more connections are needed for nodes clustering.	More computations are needed (in the server side) in order to arrange the nodes in clusters.	Very efficient dealing with non iid data, it is also ideal for scenarios where clients need to have personlized models.		Personalized models tend to perform better because they can more closely align with the statistical properties of the local data, without losing the benefits of the global model information.
FedGKD (2023)	It is expected to have the same communication costs as Fedavg, not mentioned if it has any improvement or additional cost in term of communication.	It required more computations, since the node has to teach the local model from the global model, meaning to have to use 2 models in each iteration.	Because the local models are guided by the global model, it gives a big advantage over other algorithms dealing with non iid distrbuted data over the nodes.	Yes	Like the FedMDC model, it performs the best dealing with non-iid data, while the MDC add more computation to the server side, GKD add more computation to the nodes side.
LCNSFL (2024)	Use approximation for node selection by choosing nodes closer or more directly connected to the cloud (server), reducing latency and potential data transmission costs.	It has more computational cost in the server side, but might be usfulle in the nodes side, reducing the non useful training due to connection failures.			It is mostly useful for large-scale networks with diverse geographical distributions, and conections to the server are very expensive.
Flama	Not studied or not mentioned what is the effet of this algorithm on the communication cost.		giving more impact to the 'important' data sampel, it is very effective to use this algorithm when expecting to have high level of statistical heterogeneity.		

Table 2. State-of-the-art FL algorithms comparison against the selected key factors.

As shown in Table 2, each algorithm offers unique advantages and faces potential limitations depending on the context of its deployment. By understanding these characteristics, our aim is to develop a more advanced advisory system in future studies that can guide stakeholders in selecting the optimal algorithm for their specific use case.

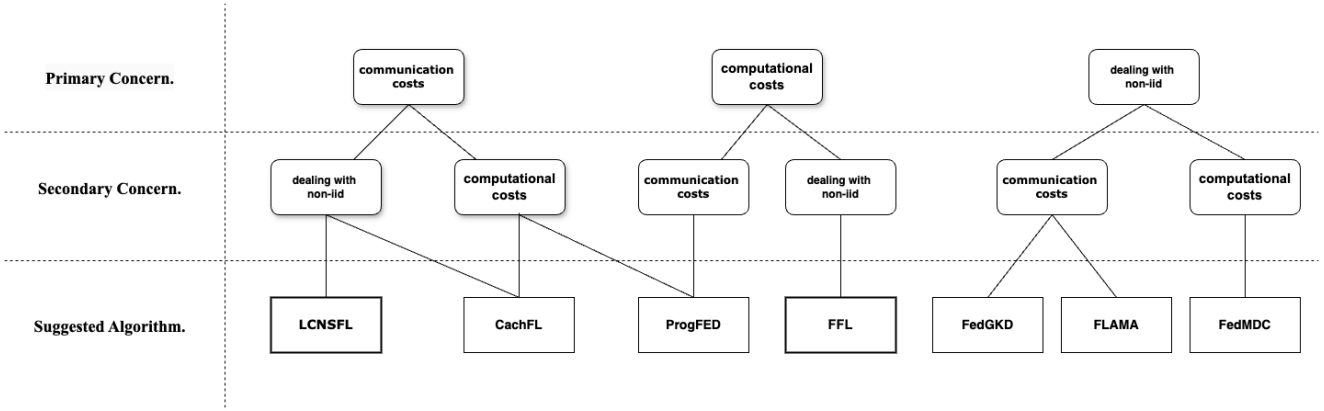


Figure 12. The proposed decision tree for selecting the best algorithm based on the sorted concerns.

8 Conclusion

The results of our study highlight the potential of advanced node selection and workload optimization strategies to enhance the performance and scalability of FL systems. By tailoring the FL process to the specific characteristics of the nodes and the data they hold, we can achieve faster convergence rates, more accurate models, more efficient use of network resources, and faster training time.

One of the key takeaways from this study is that no single algorithm is universally optimal across all FL applications. Instead, the choice of algorithm should be guided by the specific constraints and objectives of the deployment environment, including data distribution, network conditions, and the computational capabilities of participating nodes. By choosing the right FL algorithm and adapting it to the specific needs of the application, it becomes feasible to create a sustainable and efficient framework not only for COBOL but also for a wide range of future projects where federated learning can play a transformative role. This adaptability ensures that FL remains a viable solution for modern distributed systems, fostering innovation and efficiency in a variety of fields.

7 References

1- McMahan, B., Moore, E., Ramage, D., Hampson, S., & y Arcas, B. A. (2017). Communication-efficient learning of deep networks from decentralized data.

In Artificial intelligence and statistics (pp. 1273–1282).

2- Yang, Qiang, et al. "Federated machine learning: Concept and applications." ACM Transactions on Intelligent Systems and Technology (TIST) 10.2 (2019): 1-19.

3- Ye, Mang, et al. "Heterogeneous federated learning: State-of-the-art and research challenges." ACM Computing Surveys 56.3 (2023): 1-44.

4- Nori, M. K., Yun, S., & Kim, I.-M. (2021). Fast federated learning by balancing communication trade-offs. IEEE Transactions on Communications, 69(8), 5168–5182.

5- Wang, H.-P., Stich, S., He, Y., & Fritz, M. (2022). Progfed: Effective, communication, and computation efficient federated learning by progressive training. In International conference on machine learning (pp. 23034–23054).

6- Li, T., Sahu, A. K., Sanjabi, M., Zaheer, M., Talwalkar, A., & Smith, V. (2018). On the convergence of federated optimization in heterogeneous networks. arXiv preprint arXiv:1812.06127 .

7- Liu, Y., Su, L., Joe-Wong, C., Ioannidis, S., Yeh, E., & Siew, M. (2023). Cache-enabled federated learning systems. In Proceedings of the twenty-fourth international symposium on theory, algorithmic foundations, and protocol design for mobile networks and mobile computing (pp.1–11).

8- Wang, B., Lu, T., Guo, W., Chang, S., Liu, G., Huang, Q., & Yang, P. (2022). Multi-center federated learning with model decoupling. In 2022 3rd international conference on computer science and management technology (iccsmt) (pp. 450–455).

9- Yao, D., Pan, W., Dai, Y., Wan, Y., Ding, X., Yu, C., . . . Sun, L. (2023). Fedgkd: Towards heterogeneous federated learning via global knowledge distillation. IEEE Transactions on Computers.

10- Wang, W., Li, P., Li, S., Zhang, J., Zhou, Z., Wu, D. O., . . . Gong, P. (2024). Optimizing proximity strategy for federated learning node selection in the space-air-ground information network for smart cities. IEEE Internet of Things Journal, 1-1. doi: 10.1109/JIOT.2024.3416943

11- Wang, R. (2023). Federated learning with adaptive weighted model aggregation. In 2023 ieee/acm 31st international symposium on quality of service (iwqos) (p. 1-2). doi: 10.1109/IWQoS57198.2023.10188800

12- Baresi, L., Dolci, T., & Wehbe, I. (2024). On assessing heterogeneity management solutions in federated learning systems. In 4th workshop on distributed machine learning for the intelligent computing continuum (dml-icc). Retrieved from https://www.lrc.ic.unicamp.br/dml-icc/ (Inconjunction with IEEE/ACM UCC 2024, December 16–19, Sharjah, UAE)