FLASH: A framework for Federated Learning with Attribute Selection and Hyperparameter optimization

Ioannis Christofilogiannis, Georgios Valavanis, Alexander Shevtsov,
Ioannis Lamprous, and Sotiris Ioannidis

This is the author's accepted manuscript of an article published in:

2025 3rd International Conference on Federated Learning Technologies and Applications (FLTA)

The final version is available at IEEE Xplore.

Funding Acknowledgement:
This research is funded by the European Commission (Horizon Europe Programme),
under the project SYNAPSE (Grant Agreement No. 101120853).

© 2026 IEEE. Personal use of this material is permitted. Permission from IEEE must be obtained for all other uses, in any current or future media, including reprinting/republishing this material for advertising or promotional purposes, creating new collective works, for resale or redistribution to servers or lists, or reuse of any copyrighted component of this work in other works.

FLASH: A framework for Federated Learning with Attribute Selection and Hyperparameter optimization

Ioannis Christofilogiannis*†, Georgios Valavanis*†, Alexander Shevtsov*, Ioannis Lamprou§ and Sotiris Ioannidis*

*Electrical and Computer Engineering, Technical University of Crete
Chania, Crete, Greece
Email: ichristofilogian@tuc.gr, gvalavanis@tuc.gr, asevtsov@tuc.gr, sioannidis@tuc.gr

§Dienekes IKE
Heraklion, Greece
Email: ioannis.lamprou@dienekes.eu

†Equal contribution: I. Christofilogiannis worked on federated FS, G. Valavanis worked on federated HPO

***Abstract*—Federated Learning (FL) has emerged as a promising paradigm for training machine learning (ML) models on decentralized data while preserving user privacy. However, key ML optimization techniques, such as Feature Selection (FS) and Hyperparameter Optimization (HPO), are often overlooked in existing FL implementations. In this paper, we introduce FLASH, a novel framework that integrates established ML optimization methods into FL workflows. FLASH aims to reduce input data noise, enhance model accuracy, and lower model complexity. The framework enables conventional FS algorithms to work in a federated setting while maintaining the privacy of each client's data. Additionally, it supports federated HPO through collaborative parameter exploration and evaluation across clients. Experimental results demonstrate that incorporating model optimization and data noise reduction into FL can significantly improve model performance, while greatly reducing model parameter size. The FLASH source code will be publicly available upon acceptance of this paper.**

***Index Terms*—Federated Feature Selection, Federated Hyperparameter Optimization, Federated Learning, distributed systems, networks, Machine Learning.**

I. Introduction

Machine Learning (ML) has witnessed a substantial surge in popularity, driven by its remarkable success across a wide range of domains and its adaptability to diverse types of data and tasks [1]. Two critical factors contributing to the versatility of ML models are: (1) their ability to identify and eliminate irrelevant or noisy features from datasets through Feature Selection (FS) techniques, and (2) their capacity to fine-tune hyperparameters to suit the specific characteristics and requirements of each dataset and task.

As privacy and data security become increasingly important, particularly in sensitive domains such as healthcare, finance, and mobile computing, Federated Learning (FL) has emerged as a compelling alternative to centralized ML. Originally introduced by Google in 2016 [2], FL allows multiple clients (e.g., devices or institutions) to collaboratively train a shared model without directly exchanging their raw data. Instead, only model updates or parameter gradients are transmitted, thus helping to preserve user privacy and data ownership.

Despite its advantages, FL faces significant challenges that hinder its performance and adoption.

These include the inability to access raw data for centralized pre-processing and optimization, the communication overhead introduced by frequent model synchronization, and the difficulty of tuning model configurations in a distributed and privacy-constrained environment. Another critical challenge is client drift, defined as the phenomenon where local model optimization diverges from global objectives due to data heterogeneity across participating clients [3], [4]. Recent empirical studies demonstrate that even minimal drift affecting a subset of clients can significantly degrade global model performance, with documented accuracy losses under heterogeneous conditions [4]. As emphasized by [5], reducing data transfer between clients and the server is crucial not only for scalability but also for minimizing the risk of privacy leakage through side channels.

To address these limitations, we introduce FLASH-a novel framework that integrates two fundamental ML optimization techniques, Feature Selection (FS) and Hyperparameter Optimization (HPO), directly into FL workflows. Unlike existing FL frameworks that largely neglect these optimization stages, FLASH provides a unified, efficient, and privacy-preserving solution for improving model performance in decentralized environments.

FLASH enables clients to perform local FS using established methods and contribute to a global feature mask through a weighted voting process, ensuring fairness while

accounting for data heterogeneity. Simultaneously, it supports collaborative HPO using randomized search over a predefined parameter space, evaluated via stratified k-fold cross-validation performed locally on each client. These enhancements reduce model complexity, improve predictive accuracy, and lower communication costs during training.

Experimental results across multiple datasets demonstrate that FLASH significantly improves average model performance (e.g., F1 score) while also achieving substantial reductions in parameter size and network load (see Table: VII). When combined, FS and HPO yield even greater improvements in learning outcomes (see Table: V).

The key contributions of this work are as follows:

- **Feature Election Algorithm**: A novel voting-based method that enables conventional FS techniques to be adapted for federated environments, improving model performance and reducing parameter size.
- **Federated Hyperparameter Optimization (HPO)**: Implements randomized search over a configurable parameter space, with local stratified k-fold validation to evaluate model configurations while preserving data locality and privacy.

II. Background

FL is categorized by data distribution and system architecture as stated by [2]. This work focuses on the horizontal FL paradigm, which enables collaboration between parties with identical feature spaces but different sample instances [6].

In horizontal FL environments, while clients share identical feature spaces, they maintain distinct data distributions that can evolve independently over time. This architectural characteristic makes horizontal federated systems particularly vulnerable to model drift, where temporal changes in client data distributions ($P_t(X) \neq P_{t+1}(X)$) or shifts in feature-target relationships ($P_t(y|X) \neq P_{t+1}(y|X)$) can propagate through the global aggregation process and substantially degrade model performance across all participating clients [4]. These challenges are compounded by various types of data skew inherent to horizontal FL, including feature distribution skew, label distribution skew, and quantity skew [7], [8], which significantly impact aggregation algorithm performance and model convergence [9], [10]. The lack of standardized metrics for assessing data heterogeneity in federated environments further complicates the selection of appropriate handling strategies [8].

To address these challenges, FL employs different aggregation algorithms that combine local model updates from distributed clients. The foundational FedAvg algorithm [2], that we used on our experiments, uses weighted averaging based on client dataset sizes:

$$\theta_{global} = \frac{\sum_{i=1}^{N} n_i \theta_i}{\sum_{i=1}^{N} n_i}$$

where n_i represents the number of samples in client i's local dataset and θ_i represents the client's model parameters. However, FedAvg struggles with non-IID data distributions, leading to the development of more robust algorithms like FedProx [11], which adds a proximal term to limit weight divergence during local training, and SCAFFOLD [3], which uses variance reduction to mitigate client drift.

Beyond statistical challenges, FL faces significant communication overhead due to frequent model synchronization between clients and servers. As emphasized by Li et al. [5], reducing data transfer is crucial not only for scalability but also for minimizing privacy leakage through side channels. This communication bottleneck becomes particularly pronounced when model parameters are large, highlighting the need for techniques that can reduce model complexity while maintaining performance.

Traditional ML optimization techniques, such as Feature Selection (FS) and Hyperparameter Optimization (HPO), become significantly more challenging in federated environments. In centralized settings, these techniques are fundamental for identifying relevant features and optimal model configurations [12], [13]. However, the distributed nature of FL, combined with privacy constraints, prevents direct application of conventional optimization methods. FS can reduce both model complexity and communication costs by eliminating irrelevant features, while HPO ensures optimal model performance across diverse client data distributions [14]. The integration of these techniques into FL workflows represents a critical but underexplored area of research.

These data heterogeneity challenges underscore the importance of robust optimization techniques in federated environments, particularly FS and HPO, which become significantly more complex when data cannot be centralized.

III. Related Work

Federated FS addresses the challenge of selecting relevant features in distributed environments where data centralization is infeasible, ensuring privacy preservation. While this domain has received significant attention, existing approaches often propose complex, domain-specific solutions. For instance, SeiFS [15] requires specialized cryptographic circuits with customized data encoding methods, while the recently published Fed-MOFS [16] necessitates hybrid mutual information-clustering frameworks with custom multiobjective optimization. Other approaches introduce novel three-party architectures with evolutionary algorithms [17] or rely on distributed homomorphic encryption with specialized protocols [18]. These methods, while effective, require practitioners to adopt entirely new algorithmic frameworks and demand expertise in cryptography or specialized optimization techniques. In contrast, our work enables conventional feature selection methods to operate seamlessly in federated environments, reducing implementation complexity while leveraging existing domain expertise.

Federated HPO, another important component of our work, is crucial for enhancing model performance and convergence speed. It involves fine tuning hyperparameters like learning rate, which can significantly impact the model's effectiveness. The goal of HPO is to identify the optimal combination

of these hyperparameters. Grid Search and Random Search are two early foundational approaches to HPO. Grid Search systematically evaluates all combinations of predefined hyperparameter values, while Random Search samples configurations from predefined distributions according to [13]. Random Search offers several advantages over Grid Search, as shown in [13]. Firstly, it is more likely to find optimal configurations when certain hyperparameters significantly affect model performance. Secondly, it maintains statistical validity even when stopped early. And thirdly, it consistently outperforms Grid Search given the same computational resources. These advantages stem from Random Search's ability to explore the hyperparameter space more effectively, rather than exhaustively exploring the whole space. Those traits make Random Search a superior baseline approach, and that justifies its implementation in our framework.

Recent advances in HPO have introduced more sophisticated approaches. Optuna by [19] is a HPO framework that allows users to define their hyperparameter search space dynamically as they run their code. Additionally, Optuna supports advanced techniques like pruning, which helps to eliminate trials that are not performing well, making the optimization process more efficient. Another significant advancement in HPO is FedPop by [20]. This framework utilizes population-based evolutionary algorithms to enable online tuning-while-training, allowing hyperparameters to be optimized continuously during the training process. While these HPO approaches are more sophisticated than Random Search, they require integration effort, understanding of pruning/evolutionary mechanisms, and careful configuration. Random Search is trivial to implement and requires minimal setup, allowing it to be used on many different ML models and to be integrated into our framework seamlessly.

Several leading open-source FL frameworks, which have been comprehensively reviewed in a recent survey [21] include Flower [22], FedML [23], and Google's TensorFlow Federated framework [24]. Flower stands out for its ability to handle up to 15M clients using just two GPUs, supporting heterogeneous environments through language and ML framework-agnostic design. Furthermore, it is the highest scoring framework in a recent comparative study [21]. FedML offers options for on-device training and distributed computing, with standardized implementations of popular FL algorithms. TensorFlow Federated (TFF) provides an FL API but is limited to TensorFlow models.

The literature reveals two significant gaps in current federated learning research. First, existing federated FS approaches require specialized cryptographic or optimization expertise rather than enabling conventional FS methods to operate in federated environments, creating adoption barriers and preventing leveraging of established domain expertise. Second, while individual components for federated FS and HPO exist, there is a notable absence of integrated frameworks that combine both techniques. Current FL frameworks lack built-in capabilities for simultaneous FS and HPO, forcing practitioners to implement these steps separately or forgo them entirely, while the FLASH framework addresses both gaps.

IV. The FLASH framework

FLASH is implemented as a hybrid C++/Python system via Cython [25], combining Python's ML ecosystem with C++'s efficiency. It supports federated feature selection (FS) and hyperparameter optimization (HPO), and is deployed with Docker [26] and Docker Compose for scalable, reproducible experimentation.

FLASH integrates standard ML models (GNB, MLP, SGD from Scikit-learn [27]) and can be extended to other frameworks and FL aggregation algorithms. Security is provided through AES/RSA encryption for parameter exchange, using the pyca cryptography library [28] and MD5 checksums for integrity.

For aggregation, FLASH implements both unweighted averaging and FedAvg [29], where client contributions are weighted by dataset size, as well as FedProx [11] for improved convergence under heterogeneity.

$$\theta_{global_weighted} = \frac{\sum_{i=1}^{N} n_i \theta_i}{\sum_{i=1}^{N} n_i}, \theta_{global} = \frac{1}{N}\sum_{i=1}^{N} \theta_i \quad (1)$$

Additionally, it includes a residual path mechanism, where each global update combines current aggregated parameters with the previous global state:

$$\theta_{global}^{t+1} = \alpha \cdot \theta_{aggregated}^{t} + \theta_{global}^{t} \quad (2)$$

This provides memory across rounds and can boost performance; while not used in the main experiments, it is evaluated in the ablation study (VI).

A. Feature Election Algorithm

As the first aggregation step, FLASH reduces noisy features across clients by adapting conventional FS methods to a Horizontal FL setting. Each client applies an FS method supporting preference scoring, then transmits its score vector, binary mask of selected features, and local sample size. The server aggregates these using a FedAvg-style weighting [29], producing a global feature mask that is redistributed to all clients for subsequent FL rounds. Training on the reduced feature set lowers model size and communication cost (Table V).

FLASH currently integrates three complementary FS methods:

- Lasso [30], an embedded method available in Sklearn [27];
- Sequential Attention [31], combining iterative attention with greedy forward selection;
- PPFS [32], a recent wrapper-based MB-FS method available in PyImpetus.

To tune Lasso and Sequential Attention, we use Optuna with ASHA pruning [14], achieving an order of magnitude more trials within the same runtime, while PPFS requires no tuning. Clients then return the best-performing selection as

a voting vector, which the server consolidates into the final mask.

A tunable *freedom degree* parameter controls how many features from partial client agreements are retained, interpolating between strict intersection ($fd = 0$) and union ($fd = 1$). While useful for flexibility, this parameter currently requires manual tuning-a limitation we discuss further in Section VI.

Algorithm 1 Feature Election Algorithm

Require:
K clients with local datasets
Each client i has:
  - Binary mask of selected features $S_i[j]$
  - Feature preference scores $M_i[j]$
  - Number of samples n_i
freedom_degree $\in [0, 1]$
Ensure: Global feature mask G
Calculate global masks based on client selection:
$Intersection \leftarrow \bigcap_{i=1}^{K} S_i$
$Union \leftarrow \bigcup_{i=1}^{K} S_i$
if freedom_degree = 0 **then**
  $G \leftarrow Intersection$
else if freedom_degree = 1 **then**
  $G \leftarrow Union$
else
  Compute client weights:
  $w_i \leftarrow \frac{n_i}{\sum_{k=1}^{K} n_k}$ for each client i
  Initialize:
  Selected $\leftarrow \emptyset$
  Normalize preference scores:
  $M_i \leftarrow \hat{M}_i$ for each client i
  Select intersection features :
  Selected $\leftarrow$ Selected $\cup \bigcap_{i=1}^{K} S_i$
  Compute aggregated votes of all clients:
  $V \leftarrow \sum_{i=1}^{K} w_i \cdot M_i$
  Only process features that belong in the Union but not in the Intersection:
  $V_E \leftarrow V \cap (Union \setminus Intersection)$
  $x \leftarrow \lceil \text{freedom_degree} \cdot |V_E| \rceil$
  Select top x features, decided by freedom degree:
  Selected $\leftarrow$ Selected $\cup$ TopScores(V_E, x)
  $G \leftarrow$ Selected
end if
return G

B. Hyperparameter Optimization

After feature selection, FLASH performs HPO to identify the most effective model configurations for the given classification task. The server samples parameter combinations from predefined ranges in `config.json` using Random Search, ensuring broad exploration. These configurations (e.g., kernel types in SVM, loss functions in SGDC, activation functions in MLP) are distributed to clients, which evaluate them via stratified k-fold cross-validation on local data. Before evaluation, client models are aggregated so that performance reflects the global parameter state. The configuration achieving the highest validation score is selected for subsequent FL rounds.

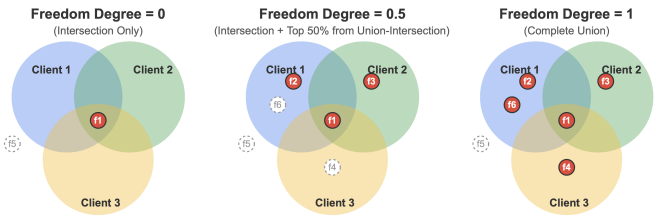

Fig. 1. Freedom degree visualization: Intersection(0.0), arbitrary value(0.5) and Union(1.0)

TABLE I
FL FRAMEWORK OVERVIEW

Feature	FLASH	Flower	FedML	TFF
Federated FS	✓	×	×	×
Built-in HPO	✓	×*	×	✓**
Cross-platform Support	✓	✓	✓	✓
Native Security Features	✓	✓	✓	✓

Compared to existing frameworks, FLASH natively integrates both FS and HPO. While extensions exist in Flower [22], [33] and TFF [24], these are either external add-ons or limited to TensorFlow models. Tables II and III show that FLASH uniquely supports both components in a plug-and-play manner.

A current limitation is that our HPO protocol has not yet been benchmarked under energy-constrained or mobile-edge scenarios, nor have communication efficiency and convergence dynamics been extensively profiled. We identify these as directions for future work to strengthen real-world applicability.

V. EXPERIMENTAL EVALUATION

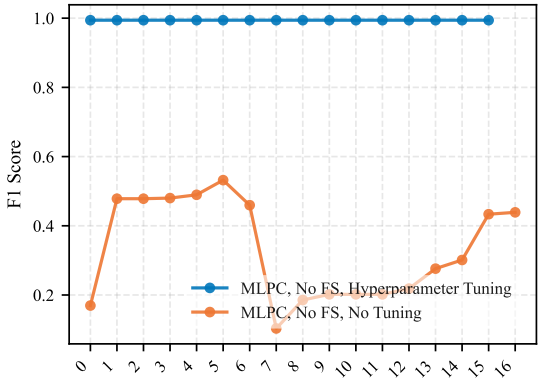

Fig. 2. MLP Performance on the breast cancer dataset with and without HPO, in a scenario without FS.

We evaluated FLASH in a horizontal FL setting with three clients, partitioning datasets unevenly (50%, 30%, 20%) using stratified splits to preserve class balance across all clients. Each client's data was further divided into training and testing sets using an 80:20 split with stratification maintained at both

TABLE II
FEDERATED LEARNING FRAMEWORK COMPARISON

Framework	Features	Interop.	Usability	Total	Org.	General Focus	Key Characteristics
Flower [22]	82.5%	100.0%	80.0%	84.8%	Adap	Research, Industrial	Framework-agnostic, supports millions of clients, easy Python API
FLARE [34]	85.0%	100.0%	70.0%	80.5%	NVIDIA	Healthcare	Enterprise-grade, strong security (HE, MPC), Docker/K8s deployment
FedScope [35]	100.0%	75.0%	67.5%	78.8%	Alibaba	Heterogeneity	Event-driven architecture, handles data and system heterogeneity, auto-tuning
FLASH	**67.5%**	**75.0%**	**75.0%**	**72.8%**	**Academic**	**Federated FS, HPO**	Feature Election and federated HPO, built using C++ and Python, Docker support
PySyft [36]	75.0%	100.0%	60.0%	72.5%	OpenMined	Privacy	Strong privacy focus, differential privacy, secure MPC, PyTorch integration
FedML [23]	82.5%	100.0%	52.5%	71.0%	FedML Inc.	Research	Three computing paradigms, mobile/IoT support, extensive benchmarks
OpenFL [37]	67.5%	37.5%	82.5%	69.0%	Intel	Enterprise	Production-ready, trusted execution environments, PKI security
EasyFL [38]	25.0%	62.5%	95.0%	67.5%	NTU	Beginners	Minimal coding required, easy and rapid prototyping
IBM FL [39]	67.5%	100.0%	52.5%	66.5%	IBM	Enterprise	Business environments, multiple ML frameworks, custom aggregator and fusion logic
TFF [24]	55.0%	37.5%	77.5%	62.8%	Google	Mobile	Open-source, TensorFlow-based, functional programming
FedLab [40]	37.5%	50.0%	77.5%	60.0%	SMILELab	FL Algorithms	Lightweight, communication efficiency, flexible APIs, PyTorch-based
FLSim [41]	60.0%	25.0%	62.5%	54.3%	Meta	Simulation	Domain-agnostic, minimal API, fast simulation, Differential Privacy support
FLUTE [42]	55.0%	27.5%	42.5%	43.3%	Microsoft	Simulation	Azure ML integration, scalable to millions, privacy features

TABLE III
COMPREHENSIVE EVALUATION OF FLASH FRAMEWORK

Evaluation Criteria	Score
Features (30%)	**67.50%**
Security Mechanisms (35%) - Cryptographic only	0.5
FL Algorithms (25%) - FedAvg and FedProx	1.0
ML Models (25%) - Sklearn and experimental PyTorch	1.0
FL Paradigms (15%) - Horizontal FL only	0.0
Interoperability (20%)	**75.00%**
Edge Device Rollout (50%) - Supported with restrictions	0.5
OS Support (25%) - Windows, MacOS, Linux	1.0
GPU Support (15%) - Fully supported	1.0
Docker Installation (10%) - Available on release	1.0
User Friendliness (50%)	**75.00%**
Development Effort (25%) - Medium effort required	0.5
Model Accuracy (25%) - 89% on MNIST	0.5
Documentation (20%) - Moderate quality	0.5
Training Speed (10%) - 11 seconds with GPU	1.0
Data Preparation (10%) - Low effort	1.0
Model Evaluation (5%) - Built-in methods	1.0
Pricing (5%) - All features free	1.0
Overall FLASH Score	**72.75%**

TABLE IV
DATASETS USED FOR EVALUATION

Dataset	Description
Mushroom Dataset [43]	Edible: 4,208 (51.8%), Poisonous: 3,916 (48.2%) *(8,124 samples, 22 features)*. Hypothetical samples of 23 gilled mushroom species, annotated by edibility. Recommended by Flower Datasets repository [22].
Adult Income Dataset [44]	$\leq$50K: 37,155 (76.1%), >50K: 11,687 (23.9%) *(48,842 samples, 14 features)*. Extracted from the 1994 US Census database for income prediction. Recommended by Flower Datasets repository [22].
Heart Disease Dataset [45]	No Heart Disease: (91.3%), Heart Disease: (8.7%) *(400k samples, 18 features)*. Contains anonymized records of patient health indicators from 2022 annual CDC survey data.
Wisconsin Diagnostic Breast Cancer Dataset [46]	Benign: 357 (62.7%), Malignant: 212 (37.3%) *(569 samples, 30 features)*. Features computed from a digitized image of a fine needle aspirate (FNA) of a breast mass.

levels. This setup reflects realistic data heterogeneity scenarios where institutions have different sample sizes while controlling for class distribution effects. All experiments systematically tested three FS methods (Lasso [30], Sequential Attention [31], and PyImpetus [32]) with and without HPO to assess the framework's adaptability across different feature selection approaches. Performance was measured by average client F1 score and total server communication cost to evaluate both predictive accuracy and communication efficiency.

a) Evaluation protocol.: We report weighted F1 scores computed as: $\overline{\text{f1}} = \frac{1}{N}\sum_{i=1}^{N} \text{f1}_i$ where f1_i is F1 score of i_{st} client. Weighted FedAvg aggregation [29] was used unless otherwise noted, as it consistently outperformed unweighted averaging except in some GNB cases.

b) Baselines and methodology.: Initial experiments without HPO established baseline performance and identified optimal freedom degree values for each dataset-model-FS combination. These were then used for Feature Election, followed by HPO. Results track both F1 improvements and parameter-size reductions (Table VII).

c) Results.: Tables V and VI show that FLASH consistently improves over baselines and often surpasses Flower [22], particularly on models sensitive to hyperparameters.

- MLP: Strong improvements (e.g., 0.41→0.99 on Breast Cancer) due to optimal tuning of learning rate, hidden layers, and activations.
- SGDC: Gains (0.02→0.45 on Income) stem from better learning rate/regularization tuning and noise reduction via FS.
- GNB: Modest improvements (e.g., 0.93→0.96) reflect its simplicity; FS helps by removing correlated features.

On average, communication costs were reduced by 49% (Table VII), with variation depending on FS method and freedom degree. This reduction both saves bandwidth and may reduce privacy leakage [5]. Figures 3-5 show that FS and HPO

TABLE V
PERFORMANCE COMPARISON ACROSS MODELS (F1-SCORES WITHOUT/WITH HPO)

Model	DS	No FS	PyImpetus	Lasso	S.Attention
GNB	M	0.81/**0.80**	0.88/**0.88**	0.87/**0.88**	0.88/**0.85**
	I	0.43/**0.45**	0.43/**0.44**	0.43/**0.42**	0.43/**0.41**
	BC	0.93/**0.94**	0.92/**0.95**	0.85/**0.96**	0.93/**0.94**
	HD	0.83/**0.79**	0.86/**0.83**	0.88/**0.86**	0.88/**0.83**
SGDC	M	0.90/**0.99**	0.90/**0.93**	0.91/**0.97**	0.86/**0.85**
	I	0.02/**0.45**	0.35/**0.45**	0.32/**0.44**	0.35/**0.39**
	BC	0.85/**0.97**	0.80/**0.98**	0.76/**0.95**	0.80/**0.97**
	HD	0.88/**0.87**	0.88/**0.87**	0.88/**0.87**	0.88/**0.87**
MLPC	M	0.90/**1.0**	0.91/**1.0**	0.92/**1.0**	0.87/**0.96**
	I	0.31/**0.47**	0.47/**0.47**	0.46/**0.46**	0.45/**0.45**
	BC	0.41/**0.99**	0.94/**0.99**	0.68/**0.96**	0.70/**0.92**
	HD	0.89/**0.89**	0.89/**0.89**	0.89/**0.89**	0.88/**0.88**

TABLE VI
PERFORMANCE COMPARISON: FLOWER VS BEST CONFIGURATION (F1-SCORES)

		No FS, No HPO	Best FLASH Configuration	
Model	DS	Flower	f1	Configuration (from Table V)
GNB	M	0.75	**0.88**	PyImpetus
	I	0.45	0.45	NoFS, HPO
	BC	**0.98**	0.96	Lasso, HPO
	HD	0.86	**0.88**	Lasso
SGDC	M	0.96	**0.99**	NoFS, HPO
	I	0.44	**0.45**	PyImpetus, HPO
	BC	**0.99**	0.98	PyImpetus, HPO
	HD	**0.89**	0.88	Any
MLPC	M	0.99	**1.0**	PyImpetus/Lasso, HPO
	I	0.47	0.47	PyImpetus, HPO
	BC	0.94	**0.99**	PyImpetus, HPO
	HD	0.89	0.89	PyImpetus/Lasso

consistently accelerate convergence compared to baselines. PyImpetus, requiring no tuning, performed reliably across settings, supporting its use as a plug-and-play FS method in horizontal FL.

VI. ABLATION STUDY

We evaluated FLASH with and without the residual path mechanism, which maintains a memory of previous global parameters by merging them with the newly aggregated ones at each round. Unlike the main experiments, these tests excluded FS and focused solely on HPO. For evaluation we report the

TABLE VII
PARAMETER SIZE REDUCTION USING FS METHODS (%)

Model	DS	PyImpetus	Lasso	S.Attention
GNB	M	53.9%	61.6%	70.8%
	I	28.6%	42.9%	28.6%
	BC	71.4%	81.0%	45.4%
	HD	43.2%	41.0%	84.3%
SGDC	M	35.2%	38.2%	35.2%
	I	6.7%	55.0%	88.8%
	BC	80.0%	77.0%	59.9%
	HD	15.8%	34.2%	52.6%
MLP	M	49.9%	38.4%	65.3%
	I	20.0%	38.7%	38.7%
	BC	63.4%	68.6%	73.8%
	HD	22.0%	26.8%	41.4%

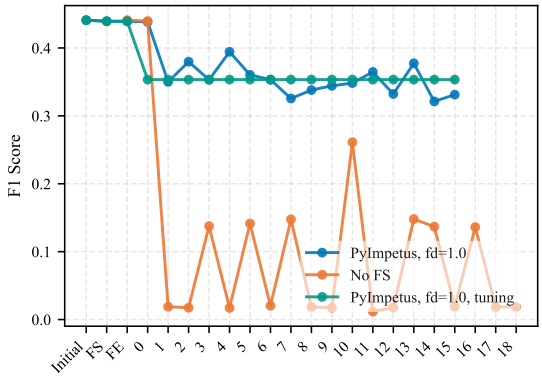

Fig. 3. SGD Performance on the income dataset with PyImpetus, freedom degree = 1.0 and HPO.

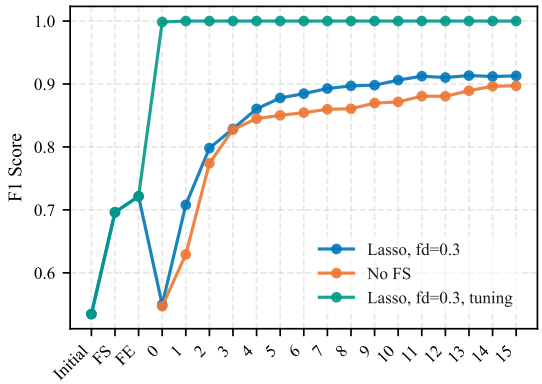

Fig. 4. MLP Performance on the mushroom dataset with Lasso, freedom degree = 0.3 and HPO.

best F1 score achieved in any round, rather than only the final round (Table VIII).

Results show that the residual path provides modest but consistent gains for simpler models, especially GNB, where the additional stability improves generalization. However, for more complex models such as MLPs, we observed no improvement and in some cases slight degradation. This suggests that the residual path may help when models have few parameters and shallow architectures, but interacts less favorably with deep optimization dynamics.

These findings indicate that while the mechanism can be useful in certain cases, it is not universally beneficial. Its impact on convergence stability, scalability under larger client populations, and behavior with heterogeneous data distributions remains an open question for future work.

VII. CONCLUSION

This paper introduces FLASH, a federated learning framework that seamlessly integrates Feature Selection (FS) and Hyperparameter Optimization (HPO), two essential but often overlooked components in FL workflows. Unlike existing frameworks such as Flower, FedML, and TFF, FLASH enables efficient model optimization while preserving data privacy and minimizing communication overhead.

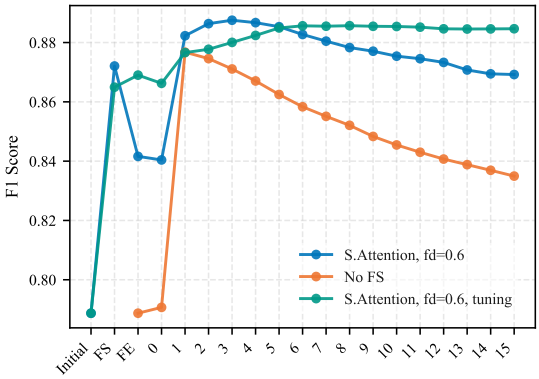

Fig. 5. GNB Performance on the heart disease dataset with S.Attention, freedom degree = 0.6 and HPO.

TABLE VIII
PERFORMANCE COMPARISON ACROSS MODELS WITH AND WITHOUT RESIDUAL PATH (BEST ACHIEVED F1 SCORES)

Model	Dataset	Base	With Residual Path
GNB	I	0.456	**0.470**
	M	**0.796**	0.792
	BC	0.947	**0.973**
	HD	0.864	**0.890**
SGDC	I	**0.451**	0.450
	M	0.998	**1.0**
	BC	**0.982**	**0.982**
	HD	0.874	**0.879**
MLP	I	**0.484**	0.483
	M	**1.0**	**1.0**
	BC	**1.0**	**1.0**
	HD	**0.887**	**0.887**

By incorporating a federated Feature Election algorithm and collaborative HPO with local cross-validation, FLASH significantly improves model performance and reduces parameter complexity. Designed to be modular and extensible, FLASH supports scalable deployment in diverse FL scenarios. The complete implementation will be made publicly available to encourage adoption, further research, and real-world application.

VIII. LIMITATIONS AND FUTURE WORK

The current implementation of FLASH, while demonstrating the benefits of integrating FS and HPO into FL, has several limitations.

First, the *freedom degree* parameter in the Feature Election algorithm is manually set, which may lead to suboptimal feature selection. Automating this parameter through adaptive election rounds or performance-based adjustment is a priority for future work. Similarly, although Optuna pruning reduces HPO cost, our framework still relies on a static configuration file, limiting runtime flexibility. We plan to implement dynamic configuration generation to support more personalized and adaptive settings. Another consideration is the absence of a formal proof for the Feature Election algorithm.

Second, our experiments focused on FedAvg [29] as a baseline aggregation strategy. While this choice allowed us to isolate the effects of FS and HPO, FedAvg is known to struggle under heterogeneous client scenarios. Future work will include testing alternative methods such as FedProx [11] - which is already implemented in FLASH and SCAFFOLD [3] to better handle client drift and system heterogeneity.

Third, the residual path mechanism showed improvements for simpler models (e.g., GNB) but was ineffective for deeper networks like MLPs. Exploring dynamic learning rates or alternative formulations may make this approach more broadly useful.

Finally, our evaluation lacks critical system-level performance metrics. We did not measure convergence rates, error variance across runs, or accuracy-communication trade-offs. Scalability with larger client populations and performance under resource-constrained environments remain unexplored, unlike production deployments that provide comprehensive device-level metrics [47], [48]. Addressing these gaps is essential to assess FLASH's real-world viability.

ACKNOWLEDGMENT

This research is funded by the European Commission (Horizon Europe Programme), under the project SYNAPSE (Grant Agreement No. 101120853).

REFERENCES

[1] J. Dean, "A golden decade of deep learning: Computing systems and applications," *Daedalus*, vol. 151, no. 2, pp. 99–116, 2022.

[2] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Aguera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics (AISTATS)*. PMLR, 2017, pp. 1273–1282.

[3] S. P. Karimireddy, S. Kale, M. Mohri, S. Reddi, S. Stich, and A. T. Suresh, "SCAFFOLD: Stochastic controlled averaging for federated learning," in *Proceedings of the 37th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, H. D. III and A. Singh, Eds., vol. 119. PMLR, 13–18 Jul 2020, pp. 5132–5143. [Online]. Available: https://proceedings.mlr.press/v119/karimireddy20a.html

[4] L. Rahimli, F. Awaysheh, S. Zubi, and S. Alawadi, "Federated learning drift detection: An empirical study on the impact of concept and data drift," 09 2024.

[5] T. Li, A. K. Sahu, A. Talwalkar, and V. Smith, "Federated learning: Challenges, methods, and future directions," *IEEE Signal Processing Magazine*, vol. 37, no. 3, pp. 50–60, 2020.

[6] Z. Zhang, W. Li, S. Wang, and Q. Yang, "Towards method of horizontal federated learning: A survey," *IEEE Transactions on Big Data*, 2022.

[7] L. Nascimento, F. M. Awaysheh, and S. Alawadi, "Data skew in federated learning: An experimental evaluation on aggregation algorithms," in *2024 2nd International Conference on Federated Learning Technologies and Applications (FLTA)*, 2024, pp. 162–170.

[8] M. Haller, C. Lenz, R. Nachtigall, F. M. Awayshehl, and S. Alawadi, "Handling non-iid data in federated learning: An experimental evaluation towards unified metrics," in *2023 IEEE Intl Conf on Dependable, Autonomic and Secure Computing, Intl Conf on Pervasive Intelligence and Computing, Intl Conf on Cloud and Big Data Computing, Intl Conf on Cyber Science and Technology Congress (DASC/PiCom/CBDCom/CyberSciTech)*, 2023, pp. 0762–0770.

[9] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-iid data," *CoRR*, vol. abs/1806.00582, 2018. [Online]. Available: http://arxiv.org/abs/1806.00582

[10] X. Ma, J. Zhu, Z. Lin, S. Chen, and Y. Qin, "A state-of-the-art survey on solving non-iid data in federated learning," *Future Generation Computer Systems*, vol. 135, pp. 244–258, 2022. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0167739X22001686

[11] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," 2020. [Online]. Available: https://arxiv.org/abs/1812.06127

[12] D. Theng and K. Bhoyar, "Feature selection techniques for machine learning: a survey of more than two decades of research," *Knowledge and Information Systems*, vol. 66, 12 2023.

[13] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of Machine Learning Research*, vol. 13, no. 2, 2012.

[14] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," in *The 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, 2019, pp. 2623–2631.

[15] R. Zhang, H. Li, M. Hao, H. Chen, and Y. Zhang, "Secure and lightweight feature selection for horizontal federated learning," *IEEE Transactions on Dependable and Secure Computing*, 2024.

[16] S. Banerjee, D. Bhuyan, E. Elmroth, and M. Bhuyan, "Cost-efficient feature selection for horizontal federated learning," *IEEE Transactions on Artificial Intelligence*, vol. 5, no. 12, pp. 6551–6565, 2024.

[17] Y. Hu, Y. Zhang, X. Gao, D. Gong, X. Song, J. Guo, and J. Wang, "A federated feature selection algorithm based on particle swarm optimization under privacy protection," *Knowledge-Based Systems*, vol. 260, p. 110122, 2023.

[18] W. Li *et al.*, "Privacy-preserving federated learning based on partial low-quality data," *Journal of Cloud Computing*, vol. 13, pp. 1–18, 2024.

[19] T. Akiba, S. Sano, T. Yanase, T. Ohta, and M. Koyama, "Optuna: A next-generation hyperparameter optimization framework," 2019. [Online]. Available: https://arxiv.org/abs/1907.10902

[20] H. Chen, D. Krompass, J. Gu, and V. Tresp, "Fedpop: Federated population-based hyperparameter tuning," 2024. [Online]. Available: https://arxiv.org/abs/2308.08634

[21] P. Riedel, L. Schick, R. von Schwerin, M. Reichert, D. Schaudt, and A. Hafner, "Comparative analysis of open-source federated learning frameworks - a literature-based survey and review," *International Journal of Machine Learning and Cybernetics*, vol. 15, no. 11, pp. 5257–5278, nov 2024. [Online]. Available: https://doi.org/10.1007/s13042-024-02234-z

[22] D. J. Beutel, T. Topal, A. Mathur, X. Qiu, J. Fernandez-Marques, Y. Gao, L. Sani, K. H. Li, T. Parcollet, P. P. B. de Gusmão, and N. D. Lane, "Flower: A friendly federated learning research framework," 2022. [Online]. Available: https://arxiv.org/abs/2007.14390

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

[24] A. Ingerman and K. Ostrowski, "Tensorflow federated," in *Proceedings of the 2nd SysML Conference*, 2019. [Online]. Available: https://www.tensorflow.org/federated

[25] S. Behnel, R. Bradshaw, C. Citro, L. Dalcin, D. S. Seljebotn, and K. Smith, "Cython: The best of both worlds," *Computing in Science & Engineering*, vol. 13, no. 2, pp. 31–39, 2011.

[26] Docker Inc. and Python Software Foundation, "Python 3.12 slim bullseye docker image," https://hub.docker.com/_/python, Docker Hub, 2024, official Python Docker image based on Debian Bullseye, tag: 3.12-slim-bullseye.

[27] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg *et al.*, "Scikit-learn: Machine learning in python," *Journal of machine learning research*, vol. 12, no. Oct, pp. 2825–2830, 2011.

[28] Python Cryptographic Authority, "cryptography: Python cryptographic library," https://cryptography.io/en/latest/, Python Cryptographic Authority, 2025, a package which provides cryptographic recipes and primitives to Python developers.

[29] H. B. McMahan, E. Moore, D. Ramage, and B. A. y Arcas, "Federated learning of deep networks using model averaging," *CoRR*, vol. abs/1602.05629, 2016. [Online]. Available: http://arxiv.org/abs/1602.05629

[30] R. Tibshirani, "Regression shrinkage and selection via the lasso," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 58, no. 1, pp. 267–288, 1996.

[31] T. Yasuda, M. Bateni, L. Chen, M. Fahrbach, G. Fu, and V. Mirrokni, "Sequential attention for feature selection," 2023. [Online]. Available: https://arxiv.org/abs/2209.14881

[32] A. Hassan, J. H. Paik, S. R. Khare, and S. A. Hassan, "A wrapper feature selection approach using markov blankets," *Pattern Recognition*, vol. 158, p. 111069, 2025. [Online]. Available: https://www.sciencedirect.com/science/article/pii/S0031320324008203

[33] J. M. Parra-Ullauri, X. Zhang, A. Bravalheri, R. Nejabati, and D. Simeonidou, "Federated hyperparameter optimisation with flower and optuna," in *The 38th ACM/SIGAPP Symposium on Applied Computing*, ser. SAC '23. New York, NY, USA: Association for Computing Machinery, 2023, pp. 1209–1216. [Online]. Available: https://doi.org/10.1145/3555776.3577847

[34] H. R. Roth, Y. Cheng, Y. Wen, I. Yang, Z. Xu, Y.-T. Hsieh, K. Kersten, A. Harouni, C. Zhao, K. Lu, Z. Zhang, W. Li, A. Myronenko, D. Yang, S. Yang, N. Rieke, A. Quraini, C. Chen, D. Xu, N. Ma, P. Dogra, M. Flores, and A. Feng, "Nvidia flare: Federated learning from simulation to real-world," 2022. [Online]. Available: https://arxiv.org/abs/2210.13291

[35] Y. Xie, Z. Wang, D. Gao, D. Chen, L. Yao, W. Kuang, Y. Li, B. Ding, and J. Zhou, "Federatedscope: A flexible federated learning platform for heterogeneity," *Proceedings of the VLDB Endowment*, vol. 16, no. 5, pp. 1059–1072, 2023.

[36] A. Ziller, A. Trask, A. Lopardo, B. Szymkow, B. Wagner, E. Bluemke, J.-M. Nounahon, J. Passerat-Palmbach, K. Prakash, N. Rose, T. Ryffel, Z. N. Reza, and G. Kaissis, *PySyft: A Library for Easy Federated Learning*, 06 2021, pp. 111–139.

[37] P. Foley, M. Sheller, B. Edwards, S. Pati, W. Riviera, M. Sharma, P. Moorthy, H. Wang, J. Martin, P. Mirhaji, P. Shah, and S. Bakas, "Openfl: the open federated learning library," *Physics in Medicine & Biology*, vol. 67, 10 2022.

[38] W. Zhuang, X. Gan, Y. Wen, and S. Zhang, "Easyfl: A low-code federated learning platform for dummies," *IEEE Internet of Things Journal*, 2022.

[39] H. Ludwig, N. Baracaldo, G. Thomas, Y. Zhou, A. Anwar, S. Rajamoni, Y. Ong, J. Radhakrishnan, A. Verma, M. Sinn, M. Purcell, A. Rawat, T. Minh, N. Holohan, S. Chakraborty, S. Whitherspoon, D. Steuer, L. Wynter, H. Hassan, S. Laguna, M. Yurochkin, M. Agarwal, E. Chuba, and A. Abay, "Ibm federated learning: an enterprise framework white paper v0.1," 2020. [Online]. Available: https://arxiv.org/abs/2007.10987

[40] D. Zeng, S. Liang, X. Hu, H. Wang, and Z. Xu, "Fedlab: A flexible federated learning framework," 2022. [Online]. Available: https://arxiv.org/abs/2107.11621

[41] L. Li, J. Wang, and C.-Z. Xu, *FLSim: An Extensible and Reusable Simulation Framework for Federated Learning*, 04 2021, pp. 350–369.

[42] M. H. Garcia, A. Manoel, D. M. Diaz, F. Mireshghallah, R. Sim, and D. Dimitriadis, "Flute: A scalable, extensible framework for high-performance federated learning simulations," 2022. [Online]. Available: https://arxiv.org/abs/2203.13789

[43] "Mushroom," UCI Machine Learning Repository, 1981, DOI: https://doi.org/10.24432/C5959T.

[44] B. Becker and R. Kohavi, "Adult," UCI Machine Learning Repository, 1996, DOI: https://doi.org/10.24432/C5XW20.

[45] K. Pytlak, "Personal key indicators of heart disease," https://www.kaggle.com/datasets/kamilpytlak/personal-key-indicators-of-heart-disease, 2022.

[46] W. Wolberg, O. Mangasarian, N. Street, and W. Street, "Breast Cancer Wisconsin (Diagnostic)," UCI Machine Learning Repository, 1993, DOI: https://doi.org/10.24432/C5DW2B.

[47] T. Yang, G. Andrew, H. Eichner, H. Sun, W. Li, N. Kong, D. Ramage, and F. Beaufays, "Applied federated learning: Improving google keyboard query suggestions," 2018. [Online]. Available: https://arxiv.org/abs/1812.02903

[48] A. Hard, K. Rao, R. Mathews, S. Ramaswamy, F. Beaufays, S. Augenstein, H. Eichner, C. Kiddon, and D. Ramage, "Federated learning for mobile keyboard prediction," *arXiv preprint arXiv:1811.03604*, 2018.