

FedAVO: Improving Communication Efficiency in Federated Learning with African Vultures Optimizer

Md Zarif Hossain^{1,2}, Ahmed Imteaj^{1,2}, Abdur R. Shahid¹

¹*School of Computing, Southern Illinois University, Carbondale, IL, USA*

²*Security, Privacy and Intelligence for Edge Devices Laboratory (SPEED Lab)*

mdzarif.hossain@siu.edu, {imteaj, shahid}@cs.siu.edu

Abstract—Federated Learning (FL) has recently experienced tremendous popularity due to its emphasis on user data privacy. However, the distributed computations of FL can result in constrained communication and drawn-out learning processes, necessitating the client-server communication cost optimization. The ratio of chosen clients and the quantity of local training passes are two hyperparameters that have a significant impact on the performance of FL. Due to different training preferences across various applications, it can be difficult for FL practitioners to manually select such hyperparameters. In this paper, we introduce FedAVO, a novel FL algorithm that enhances communication effectiveness by selecting the best hyperparameters leveraging the African Vulture Optimizer (AVO). Our research demonstrates that the communication costs associated with FL operations can be substantially reduced by adopting AVO for FL hyperparameter adjustment. Through extensive evaluations of FedAVO on benchmark datasets, we identify the optimal hyperparameters that are appropriately fitted for the benchmark datasets, eventually increasing global model accuracy by 6% in comparison to the state-of-the-art FL algorithms (such as FedAvg, FedProx, FedPSO). The code, data, and experiments have been made publicly available on our GitHub repository¹.

Index Terms—Federated Learning, African Vulture Optimizer, Communication efficiency, Local model, Global model, Convergence.

I. INTRODUCTION

The proliferation of diverse consumer electronics, including IoT devices, home appliances, smartphones, connected autonomous vehicles (CAVs), drones, Virtual Reality (VR), and Augmented Reality (AR) devices, has witnessed substantial growth in recent years. These devices capture and store various forms of data such as images, audio, and text. Image, audio, and text are just a few of the different forms of data that such devices acquire and store. Every day brings a slight increase in the number of IoT-related applications, their users, and the amount of data they produce. This phenomenon is pioneering the scope of data-driven decision-making. Due to the heterogeneous nature of these collected data, training machine learning models comes with the following challenges: *widely distributed*: data points are stored in a large number of clients, which can be significantly higher than the average number of training samples retained on a given client; *non-IID Data*: each client's data may not accurately represent the entire distribution [1] of a dataset; *unbalanced Data*: clients may hold different magnitudes of data points; *communication*

constraints: Mobile or IoT devices frequently may experience poor, expensive, or no internet connectivity; *communication cost*: data collection from edge devices is significantly high. Moreover, transferring the data to the central server for training requires additional network bandwidth.

The study FL has made substantial progress in addressing the aforementioned challenges. FL is a machine learning (ML) paradigm that trains ML models utilizing decentralized data. FL protects data privacy by preventing data transfer from local machines to a centralized server. Each edge devices train the model on its own data locally using its computation resources. Instead of sending local data to the central server, only the local updates are sent. By providing only the local updates, FL cuts down the bandwidth overhead and also the communication cost. For these distinctive qualities, FL has been adopted in a wide range of applications such as speech recognition [2], [3], surveillance system [4], [5], health care system [6], [7], Internet of Things (IoT)[8], human stroke prevention [9], [10], and cybersecurity [11]. An FL paradigm includes local training, client-server communication, and model aggregation [1], [12]. Often communication overhead in FL comes from model broadcast from the server to all clients and vice versa. Furthermore, there is a feasibility risk in every communication round in terms of constrained network bandwidth, packet transmission loss, and privacy invasion. Apart from communication overhead and privacy invasion, training on heterogeneous setups opposes some more challenges. After the centralized server broadcasts its model, the clients train the model while considering some hyperparameters such as learning rate, batch dimension, and rounds per epoch. These diversely trained client models are difficult to aggregate since computational power and data attributes (complexity, ambiguity) differ vastly between each edge device. In an idealistic scenario of hundreds, even thousands of devices, the updated global model may never converge to a global optimum due to the heterogeneous behavior of the clients. Existing aggregation techniques (e.g., FedAvg [1], FedMa [13]) only focus on integrating weights of local models. To increase generalization and convergence rate, some novel approaches suggested model aggregation by using feature fusion [14] of global and local models or using multiple global models [15].

Along with the typical hyperparameters of model training, such as learning rates, optimizers, and mini-batch sizes, FL also has particular hyperparameters, such as local epochs and participant selection. The selection of these hyperparameters can drastically affect the performance of FL. The impact

¹<https://github.com/speedlab-git/FedAVO>

Corresponding author: Ahmed Imteaj (imteaj@cs.siu.edu)

of hyperparameter tuning is much higher in Non-IID data distribution because data samples are radically different from each other across the clients. Although hyperparameter tuning optimization (HTO) has been widely explored in centralized machine learning, several aspects of HTO in FL settings yet need to be studied. In centralized machine learning, the model often trains on the entire dataset, which is often not viable in FL settings. In addition to that, models train on a wide range of hyperparameter configurations, which is extremely expensive in terms of communication and training time in FL settings because each FL cycle consists of multiple phases and communication rounds, and the model must complete each communication round in order to evaluate or change any hyperparameters. Finally, the same hyperparameter configuration functions less well as centralized machine learning due to the heterogeneous behavior of the clients and their data in FL settings. A few strategies for FL-HTO have recently been put forth; however, they concentrate on handling HTO using personalization techniques and neural networks, and they often lack in optimizing the communication cost and convergence rate. In this study, we optimize automated hyperparameter tuning with the help of a meta-heuristic algorithm. We propose FedAVO, which can configure and tune hyperparameters locally concerning the data distribution and their quality. To summarize, our proposed work outlines three novel contributions, which are listed as follows:

- To the best of our knowledge, FedAVO is the first FL hyperparameter tuning algorithm that reveals the potential of AVO as an optimizer for automated hyperparameter tuning during FL model training, specifically tailored for consumer electronics functioning as edge devices.
- FedAVO can be employed to tune the hyperparameters automatically for any state-of-the-art FL algorithm (such as FedAvg [1], FedProx [16], and FedNova [12]). Moreover, FedAVO can automatically configure and tune the hyperparameters locally on each communication rounds.
- We conduct an in-depth evaluation of FedAVO, comparing its performance with the state-of-the-art FL algorithms. Our findings reveal a notable decrease in communication overhead and an increase in convergence rate. Compared to FedAvg, FedAVO reduces the requirement of communication rounds to reach convergence by 80 on average. In addition, FedAVO converges much faster and achieves better accuracy than the state-of-the-art FL algorithms. For instance, FedAVO achieves 99.47% accuracy with MNIST Non-IID distribution and 72.64% accuracy with CIFAR-10 Non-IID distribution. Moreover, for Fashion MNIST and LISA datasets, FedAVO achieves consecutive global accuracies of 85.28% and 92.88%, respectively. The rest of the paper is organized as follows: Section II depicts the fundamental concepts of FL and the key elements of our proposed method: AVO algorithm. Related works are reviewed in Section III, while Section IV explains the algorithm and operating procedure of FedAVO. Section V presents the experimental analysis and highlights the significant findings. Finally, the study is concluded with a discussion of future directions to enhance our work in section VI.

II. BACKGROUND STUDY

African Vulture Optimization Algorithm (AVOA) is a nature inspired population based metaheuristic algorithm proposed in [17]. The algorithm was derived from the behavior and characteristics of African vultures, leveraging their adaptive and efficient foraging strategies observed in the natural world. AVOA offers a promising method for addressing complex optimization challenges across wide-range of domains. We presented the steps of the AVOA in below:

1) *Population Classification*: African vultures are categorized into three groups based on their living habits, as discussed in the cited references [18], [19]. The first group comprises vultures that represent optimal solutions when assessing their fitness with respect to feasible solutions. The second group includes vultures representing the second-best feasible solutions, while the remaining vultures are placed in the third group. Figure 1 illustrates the AVO architecture, with the right vulture symbolizing the best vulture and the left one representing the second-best. The fitness of all vulture populations is evaluated after establishing the initial population, and they are categorized based on their fitness values using Equation 1.

$$R_i^t = \begin{cases} \text{Best vulture } t_1, & \text{if } P_i^t = L_1 \\ \text{Best vulture } t_2, & \text{if } P_i^t = L_2 \end{cases} \quad (1)$$

Here, Best *vulture*₁ represents the best optimal solution for the population and Best *vulture*₂ depicts the suboptimal solution. L_1 and L_2 are both random values within the range of 0 to 1, and their sum equals to 1. With the help of the roulette-wheel technique, p_i is calculated using equation 2:

$$p_i^t = \frac{F_i^t}{\sum_{i=1}^n F_i^t} \quad (2)$$

Here, the fitness value of vultures is represented by F_i and n represents the total vultures in the first and second groups.

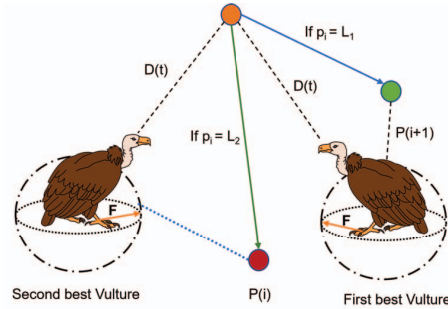


Fig. 1: Illustration of African Vulture Optimization technique.

2) *Rate of Starvation in Vultures*: In FL with AVO, we can liken vultures to computational entities or clients in the learning process. Well-fed vultures represent entities with ample resources and energy, allowing them to explore the optimization landscape extensively and perform resource-intensive computations. They handle complex tasks in Federated Learning. Conversely, starving vultures symbolize resource-constrained or energy-depleted entities, lacking capacity for prolonged and resource-intensive computations, similar to hungry vultures unable to sustain extended flights. In FL, these entities may

face limitations like restricted resources or communication bandwidth. Starvation rate among vultures significantly affects the optimization process's exploration and development phases. Changes in this rate directly impact collaboration, strategy adaptation, and computational allocation within the FL ecosystem. Ultimately, these dynamics shape the optimization process in FL with AVO. The vulture starvation rate S_i is computed by the following equation 3:

$$S_i = (2 \times \text{rand}_i + 1) \times h \times \left(1 - \frac{\text{iteration}_i}{\text{maxiterations}}\right) + t \quad (3)$$

3) *Exploration Stage*: African vultures take some time to locate other populations and food before making a long flight searching for it. They might face a hard time locating food since they spend a long time exploring their surroundings before flying farther distances in the quest of food. In AVOA, two distinct exploration strategies are discussed, and P_1 in the range of [0,1] is used to determine the strategy to be adopted. A random number rand_{p1} ranging between 0 and 1 is utilized to select one of the strategies during the exploration phase. The random value gets compared to the value of P_1 ; if the random value is lesser than P_1 , equation 4 is utilized; otherwise, equation 5 is utilized to determine the strategy:

$$P_i^{t+1} = R_i^t - D_i^t \times S \quad \text{if } P_1 \geq \text{rand}_{p1} \quad (4)$$

$$P_i^{t+1} = R_i^t - S + \text{rand}_2 \times ((ub - lb) \times \text{rand}_3 + lb) \quad \text{if } P_1 < \text{rand}_{p1} \quad (5)$$

4) *Development Stage*: In this stage, the AVO starts the first stage of exploitation, if $|S|$ is between 0.5 and 1, the vultures enter the initial development stage and during this stage, they search for prey in two strategies. P_2 ranging between 0 and 1, determines whether the vultures compete for food or spiral. round_{p2} , a random number between 0 and 1 gets generated at the first stage of this phase. If round_{p2} is greater than or equal to parameter P_2 the siege-fight strategy is applied slowly. Otherwise, the rotational flying technique is utilized. equation 6 and 7 demonstrates this procedure:

$$P_i^{t+1} = D_i^t \times (S_i + \text{rand}_4) - R_i^t - P_i^t \quad \text{if } P_2 \geq \text{rand}_{p2} \quad (6)$$

$$P_i^{t+1} = R_i^t - (Q_1^t + Q_2^t) \quad \text{if } P_2 < \text{rand}_{p2} \quad (7)$$

where, rand_4 is a random number ranging between 0 and 1, and the term $(R_i^t - P_i^t)$ is used for calculating the distance between the vulture and one of the two groups' best vultures:

$$Q_1^t = R_i^t \times \left(\frac{\text{rand}_5 \times P_i^t}{2\pi}\right) \times \cos(P_i^t) \quad (8)$$

$$Q_2^t = R_i^t \times \left(\frac{\text{rand}_6 \times P_i^t}{2\pi}\right) \times \sin(P_i^t) \quad (9)$$

5) *Final Stage (Second phase)*: The algorithm shifts to this stage when the $|S_i|$ value is less than 0.5. It indicates all vultures are full, but the best two types of vultures become hungry and weak after a prolonged flying session. At this moment, vultures attack their prey, and multiple vultures congregate at the same food source. Parameter P_3 , ranging between 0 and 1, is used to determine whether the vulture shows aggregation behavior or aggressive behavior. When the

vulture enters the second phase of the final stage, rand_{p3} gets initialized with a value between 0 and 1. When rand_{p3} is less than P_3 , the vulture engages in aggressive behavior; Otherwise, engages in aggregating behavior. This equation governs their movement and behavior during this critical phase of the optimization process. The vultures position can be updated using following equations 10, 11 and 12:

$$P_i^{t+1} = \frac{A_1^t + A_2^t}{2} \quad (10)$$

$$A_1^t = \text{Best Vulture}_1^t - \frac{\text{Best Vulture}_1^t \times P_i^t}{\text{Best Vulture}_1^t - (P_i^t)^2} \times S \quad (11)$$

$$A_2^t = \text{Best Vulture}_2^t - \frac{\text{Best Vulture}_2^t \times P_i^t}{\text{Best Vulture}_2^t - (P_i^t)^2} \times S \quad (12)$$

Here, BestVulture_1^t and BestVulture_2^t are the optimal and suboptimal solutions and S is the starvation rate.

III. RELATED WORKS

The rising prevalence of smart and IoT devices in daily life has amplified the need for FL, primarily due to its capability to safeguard data privacy during the model training process. Consequently, there has been a surge in the exploration of efficient optimization techniques tailored to FL. Researchers have directed their attention to various facets of FL, encompassing aspects such as communication efficiency [20], [21], client selection [22], [23], the intricate dynamics of statistical and system heterogeneity [24], [25] and cybersecurity [26]. One particular area of interest that aligns with our research focus revolves around the reduction of communication rounds in federated learning. Our approach to achieving this reduction centers on the optimization of hyperparameters. These hyperparameters, pivotal components of a model, govern its ability to glean insights from a specific dataset. In order to achieve better performance, the hyperparameters need to be fine-tuned with different datasets and different models. The most straightforward automated approach for adjusting hyperparameters involves conducting random searches [27] within the hyperparameter space to identify the best-performing set. However, this method was surpassed by Bayesian techniques [28], which leverage the performance of previously chosen hyperparameters to inform the selection of new ones. Notably, these methods demand substantial computational resources, as the complete training process must be executed to assess hyperparameter fitness. To mitigate the computational burden, there is a shift toward employing partial training [29], [30] with pre-selected hyperparameters. Nevertheless, even with this approach, running full or partial training to identify optimal hyperparameters within resource-constrained FL environments, where communication budgets are limited, remains prohibitively costly and cumbersome. While several researchers have explored the optimization of machine learning (ML) models and their hyperparameters using evolutionary algorithms [31], including techniques like whale optimization [32] and genetic algorithms [33], the same level of attention has not been given to FL. There have been only a few attempts to address FL HTO problems. For instance, the

FLoRA framework [34] introduces a novel HTO strategy where global hyperparameters are determined by selecting well-performing hyperparameters from local clients. FedEx [35], on the other hand, optimizes hyperparameter tuning by leveraging Neural Architecture Search (NAS) techniques with a weight-sharing architecture. Nevertheless, several optimization approaches encounter challenges in certain FL scenarios by inadvertently introducing overhead during hyperparameter tuning. Our proposed HTO method strives to bridge the existing gaps, presenting an efficient approach to optimize hyperparameters within the FL framework.

IV. PROPOSED FRAMEWORK: FEDAVO

Unlike FedAvg [1], which employs Stochastic Gradient Descent (SGD) as its optimizer, our proposed method introduces a crucial modification aimed at automating the hyperparameter tuning process for SGD. This adjustment has proven to yield substantial improvements in empirical performance, enabling faster convergence with a reduced number of communication rounds. To facilitate a better understanding of the approach, we have detailed the specific parameters and their notations in Table 1. Fig. 2 illustrates the distinct phases of FedAVO, which consist of five key stages.

Local training on client devices: In the first phase, each client performs independent local model training using the initial parameters received from the central server. They update their local model parameters by minimizing the loss function described in Equation 13 using Stochastic Gradient Descent (SGD) on their local data. Notably, this updating process occurs in complete isolation, devoid of any communication between clients or with the server. This strict isolation ensures that clients preserve the privacy of their local data and refrain from sharing it with others.

The hyperparameter tuning phase operates concurrently with the local training phase, as illustrated in Figure 2. To obtain hyperparameter tuning with AVO, an initial population of vultures is initialized to represent candidate solutions. Each vulture explores the problem space through random movement and evaluates its fitness based on an objective function. Here a problem space for any parameter x can be denoted as $x^p = \{x_u, x_l\}$. Where x_l and x_u represent the lower and upper bounds of the problem space, respectively. Vultures from the population forage through the given problem spaces for an optimal solution for a given problem. The vultures' movement is influenced by their individual experiences and interactions with other vultures within the population. Furthermore, to enhance foraging efficiency, vultures that repeatedly fail to obtain optimal solutions after a predefined number of attempts share their findings with other vultures. This iterative foraging process continues until a predetermined number of iterations or until a satisfactory solution is achieved, thereby optimizing the hyperparameters for the FL system effectively.

AVO for hyperparameter tuning: In this study, we focus on optimizing the tuning of the following SGD hyperparameters: local epoch, which is denoted as \mathcal{E} , learning rate (η), momentum (β) and weight decay, which denoted as λ . The local epoch (\mathcal{E}), which is also an FL parameter plays an important role in convergence. Performing more local epochs on

Algorithm 1: FedAVO (Proposed Algorithm)

```

1 On Server Side
2 model parameter  $\omega_o$  initialized
3 for each training round  $i = 0, 1, 2, 3 \dots$  do
4    $R_i \leftarrow$  random set of  $q$  clients
5   for each client  $k \in R_i$  do
6      $\omega_{i+1}^k \leftarrow \text{ClientUpdate}(\omega_i, k)$ 
7    $\omega_{i+1} \leftarrow \sum_{k=1}^K \frac{n_k}{n} \omega_{i+1}^k$ 
8 ClientUpdate The local client data  $\mathcal{P}_k$  is split into
   batches of size  $\mathcal{B}$ 
9 Hyperparameters  $\rho$  for the client are defined
10 The algorithm enters a loop for hyperparameter tuning
   (tuning epochs)
11 In each tuning epoch, AVO (African Vulture
   Optimization) is performed to optimize
   hyperparameters
12 A random population of vultures  $Z_i$  is initialized
13 Vultures  $Z_{vulture1}$  and  $Z_{vulture2}$  are chosen
14 For each vulture  $Z_i$ , algorithm selects a region  $R_i$ 
   based on equation (1), updates fitness value  $F$  using
   equation (2), and updates vulture's location using
   equations (4), (5), (6), (7), or (10), depending on
   conditions and probabilities ( $P_1, P_2, P_3$ )
15 The best vulture's hyperparameters  $\eta_i, \mathcal{E}_i, \lambda_i$ , and  $\beta_i$ 
   are selected
16 A loop for local training (local epochs) is initiated,
   where in each local epoch, the model parameters  $\omega$ 
   are updated using SGD
17 return  $\omega$  to server

```

clients allows more local computation and potentially reduced communication, resulting in overall global convergence. On the contrary, a larger number of local epochs may lead each device toward the optima of its local objective as opposed to the global objective due to the heterogeneous nature of clients. This may lead to slower convergence or even cause the method to diverge and overfit. Finally, we employ objective function E_k for which a problem space gets initialized with hyperparameters. The vultures forage through the problem space and search for optimal hyperparameters to solve the objective function. The optimization can be written as follows:

$$E_k(y, p) = - \sum_{c=1}^M y_{o,c} \log(p_{o,c}) \quad (13)$$

Here, 'y' represents the accurate classification for data point 'o', 'p' stands for the predicted probability for the same data point, and 'M' represents the total number of classes.

Server aggregation: In this phase, the central server aggregates the locally trained model and takes a weighted average of the local updates from the selected clients. The process is repeated until global model reaches the convergence.

Global model broadcast: Finally, the updated global model again gets sent to all selected clients and updates the local model. In the next iteration, the local data will train on the updated local model and selected hyperparameter pool by the

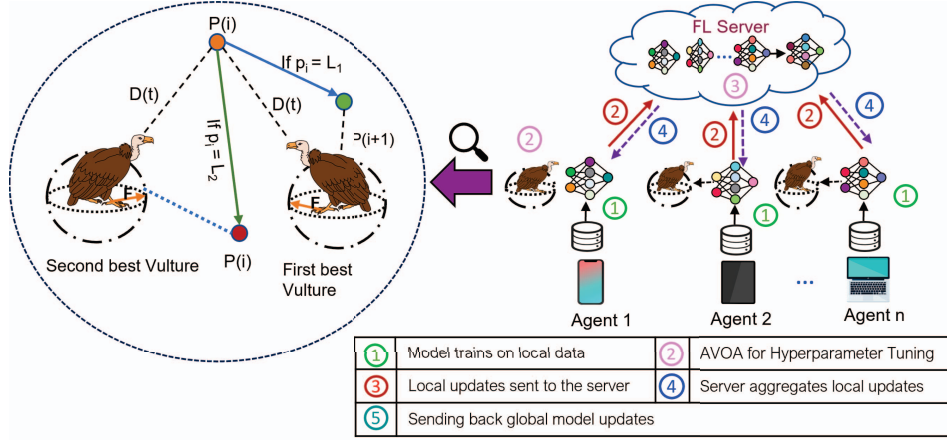


Fig. 2: System architecture of our proposed FedAVO Algorithm.

TABLE I: Key Parameters and their Notations for FedAVO

Parameter	Notation
Momentum	$\beta_i \in [\beta, \bar{\beta}]$
Local epoch	$\mathcal{E}_i \in [\mathcal{E}, \bar{\mathcal{E}}]$
Learning Rate	$\eta_i \in [\eta, \bar{\eta}]$
Weight Decay	$\lambda_i \in [\lambda, \bar{\lambda}]$
Problem Space	ρ
Population size	α

AVO. This whole process will repeat itself until a predetermined number of communication rounds or convergence is reached. Our Proposed FedAVO is presented with a complete pseudocode in algorithm 1.

V. EXPERIMENTAL EVALUATION

Our study aims to improve the convergence rate and performance of FL algorithms. In order to evaluate the proposed algorithm (FedAVO), we conduct experiments and examine the convergence speed and accuracy. Furthermore, we assess if the model achieves acceptable convergence speed and accuracy after the automated HTO with AVO compared to the baseline algorithm Vanilla FedAvg [1]. In addition, we compare our experimental results with other benchmark FL algorithms such as FedProx [16] and FedEnsemble [36].

A. Experimental Setup

We employ a High-Performance Computing Cluster (HPCC), which has two strong GPUs (each with 2,880 extra cores) and 800 CPU cores and can process up to 34 Teraflops of computations per second (one Teraflop is one million million calculations per second). In the test code, Pytorch (stable version 2.0.0) and Keras (version 2.12.0) are used. For the evaluation of FedAVO, we considered four widely used image classification datasets: CIFAR-10 [37], MNIST [38], Fashion MNIST [39], and LISA [40]. These datasets are chosen for their frequent usage in the field of image classification. For FL training, we partition these datasets into Independent and identically distributed (IID) and Non-IID settings. To tackle experimental datasets involving image data, we employ three distinct Convolutional Neural Network (CNN) models in our

initial experiment. For MNIST, we consider a simple two-layer CNN model with 28 and 64 channels, each followed by 2 x 2 maximum pooling. This model aligns with the model used to evaluate MNIST in FedAvg [1]. To ensure fairness, we match the model and parameter numbers (1,663,370 total parameters) to the FedAvg. However, for CIFAR-10 and Fashion-MNIST, we opt for three convolution layers better results. Subsequently, for training on the LISA dataset, we utilize a combination of 5 convolution layers and 3 dense layers. Since our primary focus lies in evaluating the effectiveness of our proposed hyperparameter tuning optimization method rather than achieving maximal accuracy, we find the standard CIFAR-10 model well-suited for our purposes.

TABLE II: Hyperparameter Boundaries in IID AVO Problem Space

ID	Hyperparameter	Lower Bound	Upper Bound
1	Learning Rate (η)	0.00001	0.01
2	Momentum (β)	0.1	0.9
3	Weight Decay (λ)	0.0001	0.01
4	Local epochs (\mathcal{E})	1	5

TABLE III: Hyperparameter Boundaries in Non-IID AVO Problem Space

ID	Hyperparameter	Lower Bound	Upper Bound
1	Learning Rate (η)	0.01	0.1
2	Momentum (β)	$1e^{-10}$	$1e^{-9}$
3	Weight Decay (λ)	$1e^{-10}$	$1e^{-8}$
4	Local epochs (\mathcal{E})	1	5

In this study, all our baseline algorithms (e.g., FedAvg, FedProx) and FedAVO employ the SGD optimizer. For the baseline algorithms, we keep the hyperparameters unchanged from their original study. For FedAVO, we initialize a problem space with lower bounds and upper bounds of the hyperparameters as shown in table II and table III, and we initialize the population size (α) with 50. In the training of baseline algorithms and FedAVO, we configure the batch size as 16 and the number of clients as 10. The local epoch number for baseline algorithms is standardized at 5 for each communication round. However, for FedAVO, we define the local epoch number within the problem space during the HTO. The range of hyperparameters shown in Table II and III differs based on the type of data distribution. In the case of Non-

IID distribution, a higher learning rate combined with lower momentum and weight decay proves to be more effective compared to a lower learning rate with higher momentum and weight decay. In the context of FedAVO, as discussed in Section IV, the limit for local epochs is maintained between 1 and 5 to prevent model overfitting since using a significantly higher number of local epochs would lead to overfitting of the model.

B. Experimental Results

To assess the efficacy of our FedAVO algorithm, we benchmark its performance against other state-of-the-art algorithms employing a Non-IID distribution. In our primary experiment, we divide the aforementioned datasets into 10 clients, ensuring that each client receives 500 data points containing training images. For each of our experiments, we run upto 500 FL communication rounds. Furthermore, To evaluate proposed HTO, we also compare with two state-of-the-art population-based optimizers, PSO [41] and GWO [42]. Note that, we incorporate PSO and GWO instead of AVO as an HTO naming them FedPSO and FedGWO. Subsequently, we conduct a performance comparison with FedAVO, demonstrating that AVO outperforms these population-based optimizers.

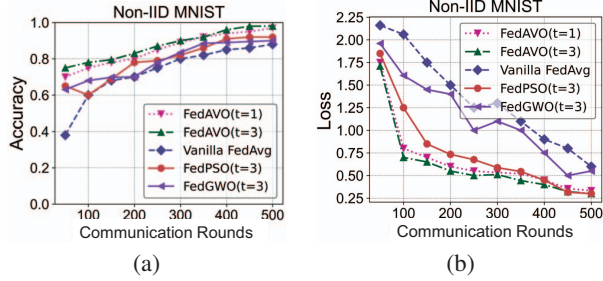


Fig. 3: Comparison of model accuracy [on left] and model loss minimization [on right] considering MNIST Non-IID setting.

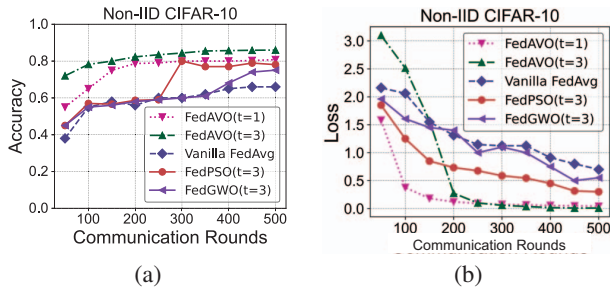


Fig. 4: Comparison of model accuracy [on left] and model loss minimization [on right] considering CIFAR-10 Non-IID setting.

In Figure 3, we evaluate our proposed FedAVO alongside the baseline methods Vanilla FedAvg [1], FedPSO, and FedGWO employing the MNIST dataset. From the comparison, the following inferences can be drawn: (1) Illustrated in Figure 3(a) and Figure 3(b), FedAVO consistently exhibits remarkable performance in terms of both global model accuracy and loss minimization in Non-IID distribution. (2) In our experiments, We explore different numbers of hyperparameter

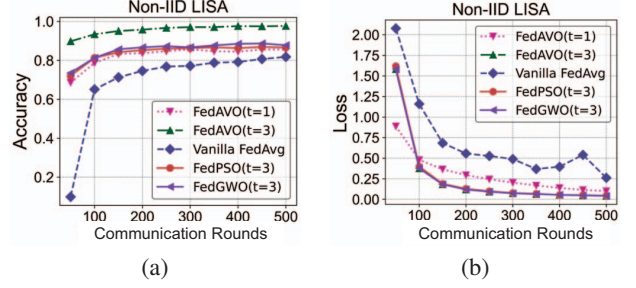


Fig. 5: Comparison of model accuracy [on left] and model loss minimization [on right] considering LISA Non-IID setting.

tuning epoch denoted as t to evaluate their respective performance. FedAVO outperforms other baseline algorithms by a considerable margin with tuning epoch 3 in terms of global model accuracy, Which demonstrates FedAVO performs better with a higher number of training epoch. However, even with 1 tuning epoch, our proposed method performs notably better than other FL algorithms as shown in Figure 3(a). Likewise, in terms of loss minimization, FedAVO with 3 epochs shows better performance compared to Vanilla FedAVG and FedGWO shown in Figure 3(b).

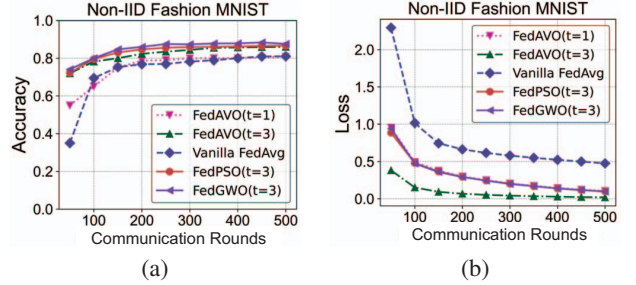


Fig. 6: Comparison of model accuracy [on left] and model loss minimization [on right] considering Fashion MNIST Non-IID setting.

Figure 4 depicts the performance comparison of FedAVO and other baseline FL algorithms on the CIFAR-10 dataset. From Figure 4(a), we perceive that FedAVO with 3 tuning epochs performs better in terms of achieving higher global model accuracy. Due to the advantage of having both local and global search techniques, FedAVO consistently performs better than FedPSO and FedGWO. As shown in Figure 4(b), Vanilla FedAvg performs poorly in terms of loss minimization. On the contrary, FedAVO minimizes loss faster than the baseline algorithms, e.g., FedPSO, FedGWO.

We also conduct empirical analysis on the LISA non-IID dataset, depicted in Figure 5(a) and 5(b). In this scenario, FedAVO and the baseline algorithms demonstrate comparable performance in terms of minimizing the loss function. However, regarding global model accuracy, FedAVO outperformed the other baseline algorithms, achieving superior results with only three hyperparameter tuning epochs. Finally, Figures 6(a) and 6(b) illustrate the performance of FedAVO on the Fashion MNIST dataset in Non-IID distribution. Notably, Both FedPSO and FedGWO show performance par with FedAVO with 3 tuning epoch. However, FedAVO shows marginal improvement in terms of loss minimization illustrated in 6(b).

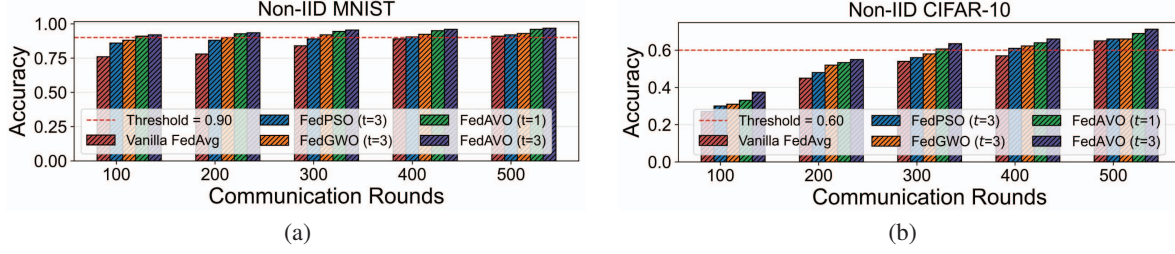


Fig. 7: Comparison of model accuracy considering (a) MNIST Non-IID and (b) CIFAR10 Non-IID setting.

TABLE IV: Global accuracy (%) of models trained by different FL algorithms on various datasets.

Dataset	Data Distribution	FEDAVG	FEDPROX	FEDENSEMBLE	FEDAVO (HP. Tuning 3 epochs)
MNIST	IID	97.70 \pm 2.07	87.49 \pm 2.05	95.85 \pm 0.68	99.67 \pm 0.15
	Non-IID	95.16 \pm 0.59	90.10 \pm 0.39	90.78 \pm 0.39	99.47 \pm 0.25
CIFAR-10	IID	67.48 \pm 0.39	76.77 \pm 0.11	77.99 \pm 0.23	76.34 \pm 0.45
	Non-IID	65.13 \pm 0.25	66.62 \pm 0.24	71.18 \pm 0.4	72.64 \pm 0.39
Fashion-MNIST	IID	84.37 \pm 0.9	84.77 \pm 0.31	83.89 \pm 0.33	89.34 \pm 0.35
	Non-IID	82.00 \pm 0.43	83.62 \pm 0.77	85.28 \pm 0.35	87.01 \pm 0.5
LISA	IID	92.48 \pm 0.39	94.24 \pm 0.5	93.43 \pm 0.25	96.34 \pm 0.4
	Non-IID	88.34 \pm 0.34	89.5 \pm 0.45	92.88 \pm 0.4	94.00 \pm 0.45

In Figure 7, we set a 90% accuracy threshold to compare FedAVO against state-of-the-art FL methods, measuring the accuracy gain over multiple communication rounds. Algorithms that surpass the threshold in fewer communication rounds are deemed to have a higher convergence rate. For the MNIST dataset, under Non-IID distribution FedAVO outperforms FedAvg by reducing communication rounds by 80. Remarkably, FedAVO achieves the specified threshold within 100 communication rounds. Furthermore, on the CIFAR-10 dataset, as depicted in Figure 7(b), FedAVO demonstrates expedited convergence. To further evaluate FedAVO, we set a 60% threshold for CIFAR-10. In contrast to FedAvg, FedAVO exhibits robust performance in CIFAR-10 Non-IID scenarios, reaching the threshold within 300 communication rounds compared to FedAvg’s 450 rounds. For the CIFAR-10 dataset, FedAVO effectively reduces communication rounds by up to 150. Based on our extensive result analysis, we can conclude that the implementation of FedAVO within FL techniques can halve or even more the communication round requirements, showcasing its efficacy in enhancing the convergence rate.

We compare our proposed FedAVO’s performance in terms of global accuracy with other state-of-the-art FL algorithms and summarize the empirical analysis in Table IV. Additionally, we showcase our empirical analysis in IID settings to provide a comprehensive overview of FedAVO’s performance. For CIFAR-10 dataset, FedAVO increases the performance gain compared to FedAvg from 67.48% to 76.34% on IID data and 65.13% to 72.64% on non-IID data. With MNIST dataset FedAVO shows a 6% increase in global accuracy compared to FedAvg. Although with CIFAR-10 IID distribution, FEDENSEMBLE shows slightly better performance by leveraging the benefit of having ensemble predictions from the user models, FedAVO shows substantially better performance on the Non-IID distribution of CIFAR-10. From the mentioned table, we can also observe that for MNIST and CIFAR-10 Non-IID distribution, FedAVO achieves 99.47% and 72.64% accuracy, respectively. Our experimentation on the Fashion-

MNIST and LISA datasets offers additional evidence supporting our claim that FedAVO consistently yields enhanced performance in terms of global accuracy. In heterogeneous systems for both of the aforementioned datasets, FedAVO shows a 5.5% increase on average in terms of global accuracy.

VI. CONCLUSION

This research focuses on reducing communication overhead in federated learning (FL) by fine-tuning the hyperparameters of consumer electronics devices involved in edge learning. FedAVO, our advanced FL algorithm, optimizes hyperparameter adjustments to significantly enhance FL performance. We conducted a thorough assessment on benchmark datasets, comparing FedAVO against state-of-the-art FL algorithms. The results unequivocally establish FedAVO’s superiority over widely adopted FL algorithms like FedAvg, FedPSO, FedEnsemble, and FedGWO, showcasing an average global accuracy improvement of up to 6%. Notably, FedAVO exhibits exceptional capabilities in reducing communication rounds, achieving an average reduction of 30 rounds for the MNIST dataset compared to FedAvg and a substantial reduction of around 150 rounds for the CIFAR-10 dataset. Even when compared to FedGWO and FedPSO, FedAVO significantly reduces communication rounds by up to 50 for CIFAR-10 and an average reduction of 20 for MNIST. Moreover, FedAVO’s adaptability extends to various FL contexts and time-sensitive systems, promising performance enhancements and reduced round requirements in practical real-world scenarios. Future research will explore further improvements in network communication performance by addressing challenges like the risk of local minima and facilitating P2P-AVO communication among clients, potentially through the implementation of dynamic multi-culture AVO.

VII. ACKNOWLEDGEMENT

This research was supported by the U.S. National Science Foundation (NSF) grant IIS-2348145. Any opinions, findings,

and conclusions or recommendations expressed in this paper do not necessarily reflect the views of the NSF.

REFERENCES

- [1] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Artificial intelligence and statistics*, pp. 1273–1282, PMLR, 2017.
- [2] M. Paulik, M. Seigel, H. Mason, D. Telaar, J. Kluijvers, R. van Dalen, C. W. Lau, L. Carlson, F. Granqvist, C. Vandeveld, *et al.*, "Federated evaluation and tuning for on-device personalization: System design & applications," *arXiv preprint arXiv:2102.08503*, 2021.
- [3] N. Tomashenko, S. Mdahaffar, M. Tommasi, Y. Estève, and J.-F. Bonastre, "Privacy attacks for automatic speech recognition acoustic models in a federated learning framework," in *ICASSP 2022-2022 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 6972–6976, IEEE, 2022.
- [4] M. Jiang, T. Jung, R. Karl, and T. Zhao, "Federated dynamic graph neural networks with secure aggregation for video-based distributed surveillance," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–23, 2022.
- [5] B. Brik, A. Ksentini, and M. Bouaziz, "Federated learning for uavs-enabled wireless networks: Use cases, challenges, and open problems," *IEEE Access*, vol. 8, pp. 53841–53849, 2020.
- [6] J. Xu, B. S. Glicksberg, C. Su, P. Walker, J. Bian, and F. Wang, "Federated learning for healthcare informatics," *Journal of Healthcare Informatics Research*, vol. 5, pp. 1–19, 2021.
- [7] R. S. Antunes, C. André da Costa, A. Küderle, I. A. Yari, and B. Eskofier, "Federated learning for healthcare: Systematic review and architecture proposal," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 13, no. 4, pp. 1–23, 2022.
- [8] A. Imteaj, U. Thakker, S. Wang, J. Li, and M. H. Amini, "A survey on federated learning for resource-constrained iot devices," *IEEE Internet of Things Journal*, vol. 9, no. 1, pp. 1–24, 2021.
- [9] C. Ju, R. Zhao, J. Sun, X. Wei, B. Zhao, Y. Liu, H. Li, T. Chen, X. Zhang, D. Gao, *et al.*, "Privacy-preserving technology to help millions of people: Federated prediction model for stroke prevention," *arXiv preprint arXiv:2006.10517*, 2020.
- [10] M. Joshi, A. Pal, and M. Sankarasubbu, "Federated learning for healthcare domain-pipeline, applications and challenges," *ACM Transactions on Computing for Healthcare*, vol. 3, no. 4, pp. 1–36, 2022.
- [11] E. Khramtsova, C. Hammerschmidt, S. Lagraa, and R. State, "Federated learning for cyber security: Soc collaboration for malicious url detection," in *2020 IEEE 40th International Conference on Distributed Computing Systems (ICDCS)*, pp. 1316–1321, IEEE, 2020.
- [12] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," *Advances in neural information processing systems*, vol. 33, pp. 7611–7623, 2020.
- [13] H. Wang, M. Yurochkin, Y. Sun, D. Papailiopoulos, and Y. Khazaeni, "Federated learning with matched averaging," *arXiv preprint arXiv:2002.06440*, 2020.
- [14] X. Yao, T. Huang, C. Wu, R. Zhang, and L. Sun, "Towards faster and better federated learning: A feature fusion approach," in *2019 IEEE International Conference on Image Processing (ICIP)*, pp. 175–179, IEEE, 2019.
- [15] K. Kopparapu, E. Lin, and J. Zhao, "Fedcd: Improving performance in non-iid federated learning," *arXiv preprint arXiv:2006.09637*, 2020.
- [16] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," *Proceedings of Machine learning and systems*, vol. 2, pp. 429–450, 2020.
- [17] B. Abdollahzadeh, F. S. Gharehchopogh, and S. Mirjalili, "African vultures optimization algorithm: A new nature-inspired metaheuristic algorithm for global optimization problems," *Computers & Industrial Engineering*, vol. 158, p. 107408, 2021.
- [18] A. H. Yakout, H. Kotb, K. M. AboRas, and H. M. Hasanien, "Comparison among different recent metaheuristic algorithms for parameters estimation of solid oxide fuel cell: Steady-state and dynamic models," *Alexandria Engineering Journal*, vol. 61, no. 11, pp. 8507–8523, 2022.
- [19] H. A. Bagal, Y. N. Soltanabad, M. Dadjuo, K. Wakil, M. Zare, and A. S. Mohammed, "Soft model parameter identification by means of modified african vulture optimization algorithm," *Energy Reports*, vol. 7, pp. 7251–7260, 2021.
- [20] D. Basu, D. Data, C. Karakus, and S. Diggavi, "Qsparse-local-sgd: Distributed sgd with quantization, sparsification and local computations," *Advances in Neural Information Processing Systems*, vol. 32, 2019.
- [21] Y. Laguel, K. Pillutla, J. Malick, and Z. Harchaoui, "A superquantile approach to federated learning with heterogeneous devices," in *2021 55th Annual Conference on Information Sciences and Systems (CISS)*, pp. 1–6, IEEE, 2021.
- [22] A. Imteaj and M. H. Amini, "Fedar: Activity and resource-aware federated learning model for distributed mobile robots," in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*, pp. 1153–1160, IEEE, 2020.
- [23] J. Xu and H. Wang, "Client selection and bandwidth allocation in wireless federated learning networks: A long-term perspective," *IEEE Transactions on Wireless Communications*, vol. 20, no. 2, pp. 1188–1200, 2020.
- [24] S. P. Karimireddy, M. Jaggi, S. Kale, M. Mohri, S. J. Reddi, S. U. Stich, and A. T. Suresh, "Mime: Mimicking centralized stochastic algorithms in federated learning," *arXiv preprint arXiv:2008.03606*, 2020.
- [25] D. Li and J. Wang, "Fedmd: Heterogeneous federated learning via model distillation," *arXiv preprint arXiv:1910.03581*, 2019.
- [26] A. R. Shahid, A. Imteaj, S. Badsha, and M. Z. Hossain, "Assessing wearable human activity recognition systems against data poisoning attacks in differentially-private federated learning," in *2023 IEEE International Conference on Smart Computing (SMARTCOMP)*, pp. 355–360, IEEE, 2023.
- [27] J. Bergstra and Y. Bengio, "Random search for hyper-parameter optimization," *Journal of machine learning research*, vol. 13, no. 2, 2012.
- [28] J. Bergstra, R. Bardenet, Y. Bengio, and B. Kégl, "Algorithms for hyperparameter optimization," *Advances in neural information processing systems*, vol. 24, 2011.
- [29] A. Klein, S. Falkner, S. Bartels, P. Hennig, and F. Hutter, "Fast bayesian optimization of machine learning hyperparameters on large datasets," in *Artificial intelligence and statistics*, pp. 528–536, PMLR, 2017.
- [30] L. Li, K. Jamieson, G. DeSalvo, A. Rostamizadeh, and A. Talwalkar, "Hyperband: A novel bandit-based approach to hyperparameter optimization," *The Journal of Machine Learning Research*, vol. 18, no. 1, pp. 6765–6816, 2017.
- [31] J.-Y. Kim and S.-B. Cho, "Evolutionary optimization of hyperparameters in deep learning models," in *2019 IEEE Congress on Evolutionary Computation (CEC)*, pp. 831–837, IEEE, 2019.
- [32] I. Aljarah, H. Faris, and S. Mirjalili, "Optimizing connection weights in neural networks using the whale optimization algorithm," *Soft Computing*, vol. 22, pp. 1–15, 2018.
- [33] X. Xiao, M. Yan, S. Basodi, C. Ji, and Y. Pan, "Efficient hyperparameter optimization in deep learning using a variable length genetic algorithm," *arXiv preprint arXiv:2006.12703*, 2020.
- [34] Y. Zhou, P. Ram, T. Salonidis, N. Baracaldo, H. Samulowitz, and H. Ludwig, "Flora: Single-shot hyper-parameter optimization for federated learning," *arXiv preprint arXiv:2112.08524*, 2021.
- [35] M. Khodak, R. Tu, T. Li, L. Li, M.-F. F. Balcan, V. Smith, and A. Talwalkar, "Federated hyperparameter tuning: Challenges, baselines, and connections to weight-sharing," *Advances in Neural Information Processing Systems*, vol. 34, pp. 19184–19197, 2021.
- [36] Z. Zhu, J. Hong, and J. Zhou, "Data-free knowledge distillation for heterogeneous federated learning," in *International Conference on Machine Learning*, pp. 12878–12889, PMLR, 2021.
- [37] A. Krizhevsky, G. Hinton, *et al.*, "Learning multiple layers of features from tiny images," 2009.
- [38] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner, "Gradient-based learning applied to document recognition," *Proceedings of the IEEE*, vol. 86, no. 11, pp. 2278–2324, 1998.
- [39] H. Xiao, K. Rasul, and R. Vollgraf, "Fashion-mnist: a novel image dataset for benchmarking machine learning algorithms," *arXiv preprint arXiv:1708.07747*, 2017.
- [40] A. Mogelmose, M. M. Trivedi, and T. B. Moeslund, "Vision-based traffic sign detection and analysis for intelligent driver assistance systems: Perspectives and survey," *IEEE transactions on intelligent transportation systems*, vol. 13, no. 4, pp. 1484–1497, 2012.
- [41] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *Proceedings of ICNN'95-international conference on neural networks*, vol. 4, pp. 1942–1948, IEEE, 1995.
- [42] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in engineering software*, vol. 69, pp. 46–61, 2014.