FedEvo: Evolutionary Population-Based Federated Learning

Minji Park, Seunghyun Yoon, Hyuk Lim
Korea Institute of Energy Technology
Naju, South Korea
{minjipark, syoon, hlim}@kentech.ac.kr

***Abstract*—Federated Learning (FL) enables collaborative model training across distributed clients without sharing raw data. In conventional FL, client updates are aggregated to construct a single global model for all participants. However, because data distributions often differ across clients, a single global model may not be optimal for every client. In this paper, we propose FedEvo, a population-based FL framework that maintains multiple candidate models on the server while preserving the standard uplink communication pattern from clients. At each round, participating clients evaluate the broadcast candidates on their local validation split, select one candidate, train it locally, and upload the resulting model update. The server then updates the population through candidate-wise aggregation, anchor-based population management, usage-based recombination, and diversity-enhancing mutation. On CIFAR-10 with Dirichlet-partitioned clients, FedEvo achieves higher accuracy than conventional FL under matched communication rounds and local optimization steps.**

***Index Terms*—Federated learning, evolutionary computation, population-based optimization.**

I. Introduction

Federated learning (FL) enables distributed clients to collaboratively train a model. The benefits of FL are that raw data is kept on-device, and only model updates are exchanged [1], [2]. The server maintains a single global model, and with each communication round, client updates are aggregated in the standard setting [1]. However, with heterogeneous clients, uneven performance across client sub-populations may occur when only one global model is used [3], [4]. Clients optimize different local objectives, and aggregated updates can pull the global model in competing directions under non-identically distributed (non-IID) data [5], [6].

Once training commits to a dominant direction, alternative candidate models that could be preferable for some clients are typically not retained. These observations motivate maintaining multiple server-side candidate models during training. A population can retain competing candidate models that are preferred by different client sub-populations and can mitigate premature commitment to a single update direction.

We propose a population-based FL framework, **FedEvo**. FedEvo maintains m candidate global models. The server broadcasts the population to the participating clients at each round. Each client evaluates the candidates and chooses one among them. Then, the server updates the population. The population is composed of stable anchors and additional candidates via exploration and exploitation. Diversity is injected into the population when client choices are concentrated on certain candidates. FedEvo maintains an uplink cost similar to that of single-model FL, while increasing the downlink and local evaluation cost linearly with m. In experiments on CIFAR-10 with Dirichlet-partitioned clients, FedEvo improves accuracy under heterogeneous data compared to FedAvg.
The main contributions of this paper are as follows:

- We propose **FedEvo**, a population-based federated learning framework with multiple server-side candidate models and standard uplink communication.
- We develop a usage-driven population management mechanism based on client selection behavior.
- We demonstrate consistent improvements over FedAvg on CIFAR-10 with Dirichlet-partitioned clients.

II. Related Work

A. Single-Model Federated Optimization under Heterogeneity

FedAvg [1] set a standard federated learning protocol by aggregating client updates, broadcasting results, and repeating the cycle. When the client data is evenly distributed, this protocol is effective and simple. However, this single-model paradigm suffers under non-IID environments. The critical problem under non-IID settings is the *client drift* issue. Learning can cancel out and destabilize useful specializations if the updates are averaged, and the local model drifts toward its population gradient. Several calibration methods work in a single-model paradigm. FedProx [3] limits each client so that the degree of deviation from the global model can be controlled. A constraint is imposed by a proximal penalty function on each local objective. SCAFFOLD [4] takes a different approach. A *control variate* is used to track the misalignment between local and global gradients. With this control variate, client drift is measured and taken into consideration during estimation. FedNova [7] normalizes local updates to address objective inconsistency under heterogeneous local training, and FedDyn [8] introduces a round-adaptive dynamic regularizer. The above methods remarkably reduce client drift. However, these methods also share the limitation of relying on a single global model throughout all learning stages.

B. Multi-Model and Personalized Federated Learning

If a single-model approach cannot satisfy all heterogeneous clients, multi-model learning is introduced. A clustered FL strategy learns a few global models and assigns each client to

one of them either explicitly or implicitly. IFCA [9] alternates between cluster assignment and cluster model update. Related approaches formulate groups based on similarity within updates or gradients [10]. Mixture and latent-variable formulations such as federated expectation maximization (EM) [11], [12] treat heterogeneity probabilistically, assigning memberships via alternating optimization. Here, the membership is not fixed; it is represented as a probability distribution.

Personalized FL is a complementary direction. Instead of grouping clients, it gives each client its own model while coupling it with a shared reference. Ditto [13] optimizes a balance between fairness and robustness across the global and local models. In this way, per-client parameters are kept on-device and are not uploaded to the server.

C. Evolutionary and Population-Based Learning

Population-based methods maintain a pool of candidate solutions and perform exploitation and exploration. Population-Based Training (PBT) [14] is a useful tool in deep learning, as it enables joint tuning of hyperparameters and weights across concurrent training runs. Neuroevolution develops this idea further. By using selection and variation as the main exploration tools, it sometimes entirely replaces the gradient. The outcome is competitive among networks with millions of parameters [15].

Federated learning has adopted these ideas in several distinct ways. EvoFed [16] replaces gradients with an evolutionary strategy (ES). Instead of sending the full model updates, it sends only the scalar score for each perturbation. In CIFAR-10, over 99% compression is achieved; however, this comes with increased local computation as well as reduced gradient fidelity. Custode et al. propose NEvoFed [17], a neuroevolution scheme that allows *heterogeneous* network architectures across clients. Each client refines its own topology. Aggregation is implicitly performed via exchange of the local models with the best performance. Follow-up work on NEvoFed suggests [18] eliminating model-parameter exchange while using only fitness signals in FL, naming this a model-free communication paradigm. On the variation-operator side, FedMR [19] recombines client models and thus suggests a new global hypothesis. FedMut [20] seeks a way out of local optima by injecting stochastic mutation into the global model. AdaFed [21] monitors client performance and aggregates local models by weighted averaging. In other words, better-performing models are given more weight. The goal is to construct one global model that generalizes well across all circumstances. Although AdaFed may not be evolutionary in the strict sense, it shares the notion that feedback should guide the optimization process.

FedEvo is positioned at the intersection of multi-model FL and population-based learning. Unlike clustered or personalized FL, it does not require fixed client grouping or client-specific model branches. Unlike evolutionary FL methods that replace standard gradient-based communication, FedEvo preserves the standard uplink pattern while maintaining and evolving multiple server-side candidates.

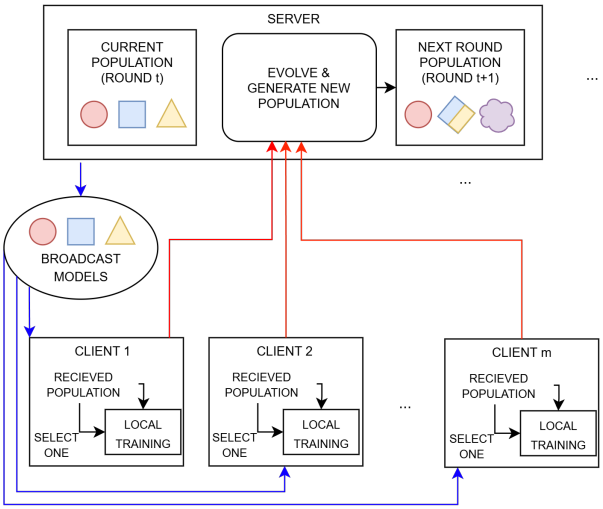


Fig. 1: Overall mechanism for FedEvo

III. FedEvo: Method

A. Overview

We consider a standard synchronous federated learning setting with a central server and a population of N clients, denoted by $\mathcal{C} = \{1, \ldots, N\}$. At communication round t, the server samples a subset of participating clients $\mathcal{S}_t \subseteq \mathcal{C}$. Each client k has disjoint local splits D_k^{tr} and D_k^{val} for training and validation, respectively.

Unlike standard single-model FL, FedEvo maintains a population of m candidate global models on the server:

$$\mathcal{P}^{(t)} = \{\theta_1^{(t)}, \ldots, \theta_m^{(t)}\}.$$

At each round, the server broadcasts the population to the selected clients. Each participating client evaluates the candidate models on its local validation split, selects one candidate, performs local training from the selected model, and uploads a single model delta. The server then aggregates updates separately for each candidate and uses the resulting usage statistics to generate the next population. Figure 1 illustrates the overall mechanism of FedEvo. Table I summarizes the main notation used throughout the description of FedEvo.

B. Client Selection and Local Update

Given a candidate model θ, client k evaluates it using the empirical validation loss

$$\mathcal{L}_k^{\text{val}}(\theta) = \frac{1}{|D_k^{\text{val}}|} \sum_{(x,y)\in D_k^{\text{val}}} \ell(f_\theta(x), y),$$

where f_θ is the network and ℓ is the task loss (e.g., cross-entropy). The selected candidate index is

$$j^*(k) = \arg\min_{j\in\{1,\ldots,m\}} \mathcal{L}_k^{\text{val}}(\theta_j^{(t)}). \tag{1}$$

If multiple candidates yield the same loss, one is chosen randomly.

TABLE I: Summary of notation used in FedEvo.

Symbol	Description
m	population size (number of server-side candidates)
$\theta_j^{(t)}$	candidate j on the server at round t
$j^*(k), \Delta\theta_k^{(t)}$	selected candidate index and uploaded model delta
$S_j^{(t)}$	set of clients selecting candidate j at round t
$c_j^{(t)}$	usage count, $c_j^{(t)} = \lvert S_j^{(t)}\rvert$
$\bar{\theta}_j^{(t)}$	candidate after candidate-wise aggregation (Eq. 3)
$p_j^{(t)}$	normalized usage, $p_j^{(t)} = c_j^{(t)} / \sum_\ell c_\ell^{(t)}$
$H^{(t)}$	entropy of the usage distribution (Eq. 4)
$\tilde{p}_j^{(t)}$	smoothed parent-sampling probability (Eq. 8)
$\theta_{\text{base}}^{(t)}$	server base reference model
$\theta_{\text{stab}}^{(t)}$	stabilizer anchor (Eq. 5)
$\theta_{\text{used}}^{(t)}$	used-index anchor (Eq. 6)
$\theta_{\text{topk}}^{(t)}$	top-k usage-weighted anchor (Eq. 7)
ρ, γ	top-k ratio and weighting exponent
$[\lambda_{\text{low}}, \lambda_{\text{high}}]$	interpolation range
τ_{factor}	entropy threshold factor for mutation (Eq. 13)
$\sigma_{\text{mut}}, \phi, \zeta, r_0$	mutation scale, mutated-layer fraction, rare-candidate fraction, and unchanged-prefix size

After selection, client k starts from $\theta_{j^*(k)}^{(t)}$ and performs local SGD on D_k^{tr} for E local epochs, obtaining a locally trained model θ_k^{local}. The uploaded delta is

$$\Delta\theta_k^{(t)} = \theta_k^{\text{local}} - \theta_{j^*(k)}^{(t)} \in \mathbb{R}^P, \tag{2}$$

where P is the number of trainable parameters. Thus, each client uploads only one model delta per round, similarly to standard FL.

C. Candidate-wise Aggregation and Usage Signal

For candidate j, let

$$S_j^{(t)} = \{k \in \mathcal{S}_t \mid j^*(k) = j\}$$

denote the set of clients that selected candidate j. The post-trained version of candidate j is obtained by aggregating the deltas from the clients in $S_j^{(t)}$:

$$\bar{\theta}_j^{(t)} = \theta_j^{(t)} + \frac{1}{\sum_{k\in S_j^{(t)}} n_k} \sum_{k\in S_j^{(t)}} n_k \, \Delta\theta_k^{(t)}, \tag{3}$$

where $n_k = \lvert D_k^{\text{tr}}\rvert$. If $S_j^{(t)} = \varnothing$, we set $\bar{\theta}_j^{(t)} = \theta_j^{(t)}$.

Client choices also induce a usage signal. Let

$$c_j^{(t)} = \lvert S_j^{(t)}\rvert, \qquad p_j^{(t)} = \frac{c_j^{(t)}}{\sum_{\ell=1}^m c_\ell^{(t)}},$$

with the convention $p_j^{(t)} = 1/m$ if $\sum_{\ell=1}^m c_\ell^{(t)} = 0$. The corresponding selection entropy is

$$H^{(t)} = -\sum_{j=1}^m \hat{p}_j^{(t)} \log \hat{p}_j^{(t)}, \qquad \hat{p}_j^{(t)} = \max(p_j^{(t)}, 10^{-12}). \tag{4}$$

These usage statistics are used to construct the next population.

IV. Detailed Population Management

After candidate-wise aggregation, FedEvo constructs the next population using the usage statistics derived from client choices. The server preserves representative anchor models, generates additional candidates through recombination, and injects diversity when client selections become overly concentrated. In this way, population management in FedEvo balances exploitation of frequently selected candidates with exploration of alternative directions.

A. Anchor Models

FedEvo always includes three anchor models in the next population. The first is the stabilizer anchor, defined from the base-referenced average client update. Let $\theta_{\text{base}}^{(t)}$ denote the server base reference model. For each client message, define the base-referenced delta

$$\Delta\theta_{k,\text{base}}^{(t)} = \theta_k^{\text{local}} - \theta_{\text{base}}^{(t)}.$$

Its weighted average over participating clients is denoted by $\bar{\Delta\theta}_{\text{base}}^{(t)}$, and the stabilizer anchor is

$$\theta_{\text{stab}}^{(t)} = \theta_{\text{base}}^{(t)} + \bar{\Delta\theta}_{\text{base}}^{(t)}. \tag{5}$$

We then update $\theta_{\text{base}}^{(t+1)} \leftarrow \theta_{\text{stab}}^{(t)}$.

Let $\mathcal{U}^{(t)} = \{j : c_j^{(t)} > 0\}$ denote the used candidate indices. The used-index anchor is

$$\theta_{\text{used}}^{(t)} = \begin{cases} \theta_{\text{stab}}^{(t)}, & \text{if } \mathcal{U}^{(t)} = \varnothing, \\ \frac{1}{\lvert\mathcal{U}^{(t)}\rvert} \sum_{j\in\mathcal{U}^{(t)}} \bar{\theta}_j^{(t)}, & \text{otherwise.} \end{cases} \tag{6}$$

Finally, let $k_{\text{top}} = \max(1, \lfloor \rho m \rfloor)$ and let $\mathcal{T}^{(t)}$ be the indices of the top-k_{top} candidates with respect to $\{c_j^{(t)}\}_{j=1}^m$. The top-k anchor is

$$\theta_{\text{topk}}^{(t)} = \sum_{j\in\mathcal{T}^{(t)}} \frac{(c_j^{(t)} + 10^{-8})^\gamma}{\sum_{\ell\in\mathcal{T}^{(t)}} (c_\ell^{(t)} + 10^{-8})^\gamma} \, \bar{\theta}_j^{(t)}. \tag{7}$$

B. Interpolation Recombination

To bias recombination toward frequently selected candidates while preserving exploration, FedEvo uses a smoothed parent-sampling distribution

$$\tilde{p}_j^{(t)} = \frac{p_j^{(t)} + \varepsilon}{\sum_{\ell=1}^m \big(p_\ell^{(t)} + \varepsilon\big)}, \tag{8}$$

where $\varepsilon > 0$.

Using this distribution, FedEvo generates additional candidates by interpolation between two parent candidates:

$$\theta_{\text{mix}} = \lambda\, \bar{\theta}_p^{(t)} + (1-\lambda)\, \bar{\theta}_q^{(t)},$$
$$p \sim \text{Cat}\Big(\tilde{p}^{(t)}\Big), \quad q \sim \text{Cat}\Big(\tilde{p}^{(t)}\Big), \quad \lambda \sim \text{Unif}[\lambda_{\text{low}}, \lambda_{\text{high}}]. \tag{9}$$

By default, we allow $p = q$; optionally, q can be resampled until $q \neq p$ to avoid generating an offspring identical to its parent.

C. Cosine-based Diversity Injection

After warm-up, FedEvo can optionally generate an injected candidate. Let $\mathrm{vec}(\cdot)$ denote parameter flattening. We first select a candidate that is most dissimilar to the stabilizer anchor in cosine similarity:

$$j_{\mathrm{anti}} = \arg\min_{j\in\{1,\dots,m\}} \frac{\langle \mathrm{vec}(\bar{\theta}_j^{(t)}),\, \mathrm{vec}(\theta_{\mathrm{stab}}^{(t)})\rangle}{\|\mathrm{vec}(\bar{\theta}_j^{(t)})\|\,\|\mathrm{vec}(\theta_{\mathrm{stab}}^{(t)})\| + \varepsilon}. \quad (10)$$

Let the weighted mean client delta be

$$\bar{\Delta\theta}_{\mathrm{cand}}^{(t)} = \frac{1}{\sum_{k\in\mathcal{S}_t} n_k} \sum_{k\in\mathcal{S}_t} n_k\, \Delta\theta_k^{(t)}. \quad (11)$$

The injected candidate is then

$$\theta_{\mathrm{inj}}^{(t)} = \bar{\theta}_{j_{\mathrm{anti}}}^{(t)} + \bar{\Delta\theta}_{\mathrm{cand}}^{(t)}. \quad (12)$$

D. Warm-cache Seeding

If there are remaining slots, FedEvo can optionally create new candidates from cached recent base-referenced client deltas. A seeded candidate is written as

$$\theta_{\mathrm{seed}}^{(t)} = \theta_{\mathrm{stab}}^{(t)} + \mathrm{Avg}(\{\Delta_k^{\mathrm{cache}}\}_{k\in\mathcal{G}}),$$

for a sampled client group $\mathcal{G}$, with additional small noise to reduce duplication.

E. Mutation Trigger and Rare-candidate Mutation

Mutation is triggered when the normalized selection entropy falls below a threshold:

$$h^{(t)} = \frac{H^{(t)}}{\log m}, \qquad h^{(t)} < \tau_{\mathrm{factor}}. \quad (13)$$

When triggered, FedEvo identifies rarely selected candidates and applies layer-wise Gaussian perturbations to a subset of parameter tensors. Specifically, let $\mathcal{R}^{(t)} = \mathrm{Bottom}(r)$ with $r = \max(1, \lfloor \zeta m \rfloor)$ with respect to $\{c_j^{(t)}\}$ denote the rarely selected candidates. For each mutated model, a subset of parameter tensors is selected, and Gaussian perturbation is applied to each selected tensor w as

$$w \leftarrow w + \mathcal{N}\big(0, (\sigma_{\mathrm{mut}} \cdot \max(\mathrm{Std}(w), 10^{-4}))^2 I\big).$$

F. Assembly of the Next Population

The next population $\mathcal{P}^{(t+1)}$ is constructed from the three anchor models, interpolation offspring, injected candidates (when enabled), optional warm-cache seeds, and additional offspring if needed. After truncation to size m, mutation is applied when triggered. Among the resulting candidates, $\theta_{\mathrm{stab}}^{(t)}$ is used as the representative model for evaluation. The complete server routine is summarized in Algorithm 1.

Algorithm 1 Server routine for FedEvo at round t

Require: Population $\mathcal{P}^{(t)} = \{\theta_j^{(t)}\}_{j=1}^m$, base reference $\theta_{\mathrm{base}}^{(t)}$, hyperparameters Θ, optional warm-delta cache

Sample participants $\mathcal{S}_t$ and broadcast $\mathcal{P}^{(t)}$
Receive $\{(j^*(k), \Delta\theta_k^{(t)}, n_k)\}_{k\in\mathcal{S}_t}$ and compute usage counts $\{c_j^{(t)}\}_{j=1}^m$
Compute aggregated candidates $\{\bar{\theta}_j^{(t)}\}_{j=1}^m$ using Eq. (3)
Compute $p^{(t)}$, $H^{(t)}$, $h^{(t)}$, and $\tilde{p}^{(t)}$ using Eqs. (4) and (8)
Compute anchors $\theta_{\mathrm{stab}}^{(t)}$, $\theta_{\mathrm{used}}^{(t)}$, and $\theta_{\mathrm{topk}}^{(t)}$ using Eqs. (5)–(7); set $\theta_{\mathrm{base}}^{(t+1)} \leftarrow \theta_{\mathrm{stab}}^{(t)}$
Initialize $\mathcal{P}^{(t+1)} \leftarrow [\theta_{\mathrm{stab}}^{(t)}, \theta_{\mathrm{used}}^{(t)}, \theta_{\mathrm{topk}}^{(t)}]$
Add n_{interp} interpolation offspring using Eq. (9)
if $t > T_{\mathrm{no\_inj}}$ **then**
    Add n_{inj} injected candidates using Eqs. (10) and (12)
end if
Fill remaining slots up to size m using warm-cache seeding and additional interpolation
if $t > T_{\mathrm{no\_mut}}$ **and** $h^{(t)} < \tau_{\mathrm{factor}}$ **then**
    Mutate rarely selected candidates (bottom ζ fraction), keeping the first r_0 candidate indices unchanged
end if
return $\mathcal{P}^{(t+1)}$ and representative model $\theta_{\mathrm{stab}}^{(t)}$

V. Experiments

We evaluate FedEvo on CIFAR-10 under controlled client heterogeneity and compare it with FedAvg as the standard single-model FL baseline. We also include FedDyn, a strong single-model method for non-IID federated optimization, and FedMut, a mutation-based approach related to population-style model updates. These baselines allow us to compare FedEvo against both standard and diversity-oriented federated learning methods.

A. Experimental Setup

1) Dataset and model.: All methods use ResNet-18 as the backbone model and are evaluated on CIFAR-10. Test accuracy is reported on the standard test split.

2) Client heterogeneity.: To control data heterogeneity, we partition the training data across clients using a Dirichlet prior with concentration parameter $\alpha \in \{0.1, 1, 10\}$. Unless otherwise specified, we use $N = 100$ clients, sample $K = 10$ participating clients per round uniformly without replacement, and run training for $T = 1000$ communication rounds.

3) Local training and validation.: Each participating client performs 5 local epochs per round. For FedEvo, each client splits its local data into disjoint training and validation sets, denoted by D_k^{tr} and D_k^{val}, with validation ratio 0.1. Model updates are computed using only D_k^{tr}, while D_k^{val} is used only for candidate selection.

4) Optimization and evaluation.: FedEvo, FedAvg, and FedMut use SGD with momentum 0.9, weight decay 5×10^{-4}, and an initial learning rate of 0.01, which decays by a factor of 0.998 per round. FedDyn is evaluated on the same dataset and under the same client partitioning setup, but uses SGD without momentum following its original implementation. The client batch size for local training is 50, and the batch size for test evaluation is 200.

5) Runs and BatchNorm recalibration.: For each α, we conduct five trials using different training seeds and report the mean $\pm$ standard deviation over the last 50 rounds. During evaluation, BatchNorm statistics are recalibrated using client-local training batches without updating model weights. The

TABLE II: FedEvo hyperparameters used in the experiments. Parameters marked with † are explicitly set in the run script, and the others follow the default `GAConfig` settings.

Parameter	Value	Description
$m^{\dagger}$	10	Population size
ρ	0.3	Top-k ratio
γ	1.5	Top-k weighting exponent
$[\lambda_{\text{low}}, \lambda_{\text{high}}]$	[0.4, 0.6]	Interpolation range
$n^{\dagger}_{\text{interp}}$	4	Number of interpolation offspring per round
n_{inj}	1	Number of injected candidates per round
$T_{\text{no-inj}}$	20	Injection warm-up rounds
$T_{\text{no-mut}}$	50	Mutation warm-up rounds
τ_{factor}	0.8	Entropy threshold factor for mutation
σ_{mut}	0.01	Mutation noise scale
ϕ	0.33	Mutated-layer fraction
ζ	0.2	Rare-candidate fraction
r_0	1	Number of rare candidates kept unchanged
σ_{init}	0.001	Initialization noise scale

same recalibration procedure is applied to both FedAvg and FedEvo.

6) FedEvo hyperparameters.: Unless otherwise specified, FedEvo uses population size $m = 10$, and $\theta^{(t)}_{\text{stab}}$ is used as the representative model for evaluation. Table II lists the FedEvo hyperparameters. Values marked with † are explicitly set in the run script, and the others follow the default `GAConfig` settings. Cosine-based injection is enabled after the warm-up period $T_{\text{no-inj}}$.

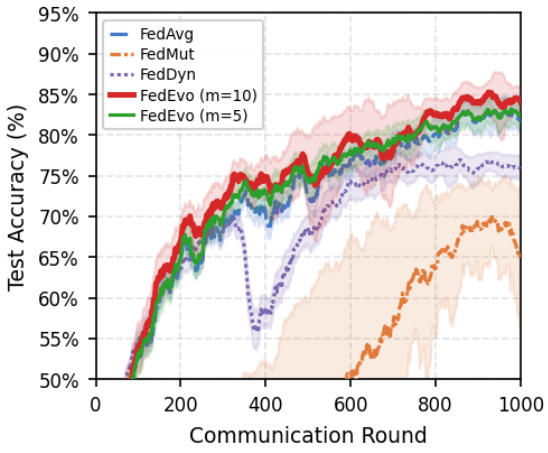
B. Results

Table III reports the final test accuracy under Dirichlet-partitioned client data. The most important observation is that FedEvo with $m = 10$ achieves the best performance across all three heterogeneity settings. In particular, FedEvo ($m = 10$) improves over FedAvg by 1.69 points at $\alpha = 0.1$, 1.13 points at $\alpha = 1$, and 1.27 points at $\alpha = 10$. These results support the main claim of this paper: maintaining and evolving multiple server-side candidates can improve federated optimization under heterogeneous client data.

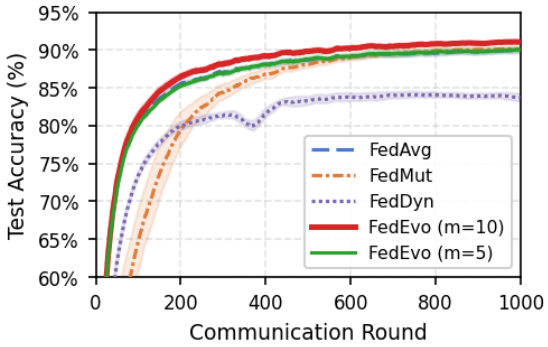
A comparison between $m = 5$ and $m = 10$ further shows that population size is important. FedEvo with $m = 5$ provides only marginal improvement over FedAvg and does not consistently outperform the stronger baselines. In contrast, FedEvo with $m = 10$ yields clear and consistent gains across all three α values. This result suggests that a small population is not sufficient to preserve useful alternative candidate models, whereas a larger population provides a more effective basis for usage-driven population management.

Among the baselines, FedDyn performs worse than FedAvg in all three settings. FedMut shows competitive performance in the moderate and near-IID settings ($\alpha = 1$ and $\alpha = 10$), but degrades substantially in the strongly non-IID case ($\alpha = 0.1$). By comparison, FedEvo remains consistently strong across all three heterogeneity levels, indicating that its population management strategy is more robust to varying client distributions.

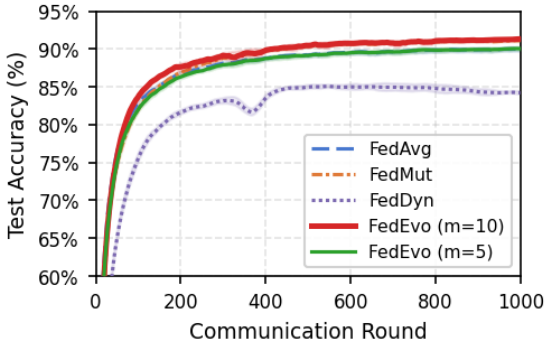
Figure 2 shows that this performance advantage is maintained in the later communication rounds, rather than arising only from transient peaks.



(a) $\alpha = 0.1$



(b) $\alpha = 1$



(c) $\alpha = 10$

Fig. 2: Test accuracy curves of the baselines (FedAvg, FedMut, and FedDyn) and FedEvo ($m = 5, 10$) under Dirichlet-partitioned client data with different heterogeneity levels: (a) $\alpha = 0.1$, (b) $\alpha = 1$, and (c) $\alpha = 10$.

VI. Conclusion

This paper presented **FedEvo**, a population-based federated learning framework for heterogeneous client data. FedEvo maintains multiple server-side candidate models, lets each client select the most suitable candidate based on local validation loss, and updates the population through candidate-wise

TABLE III: Test accuracy (mean $\pm$ std, %) over the last 50 communication rounds under Dirichlet-partitioned client data, averaged over five random seeds. FedEvo reports the representative model θ_{stab} with BatchNorm recalibration.

Algorithm	**Dirichlet α**		
	$\mathbf{0.1}$ (strong non-IID)	$\mathbf{1}$ (moderate non-IID)	$\mathbf{10}$ (near IID)
FedAvg [1]	82.53 ± 0.80	89.92 ± 0.28	89.99 ± 0.26
FedDyn [8]	76.14 ± 1.11	83.83 ± 0.44	84.23 ± 0.11
FedMut [20]	67.62 ± 4.49	90.13 ± 0.24	91.08 ± 0.19
FedEvo (m=10)	$\mathbf{84.22 \pm 1.08}$	$\mathbf{91.05 \pm 0.20}$	$\mathbf{91.26 \pm 0.09}$
FedEvo (m=5)	82.80 ± 0.73	89.98 ± 0.36	90.02 ± 0.22

aggregation and usage-driven population management. Its key idea is that client selection behavior can serve as an implicit signal for population evolution, enabling anchor preservation, recombination, and diversity control without explicit fitness reporting.

Experiments on CIFAR-10 with Dirichlet-partitioned clients showed that FedEvo consistently improves over FedAvg across multiple heterogeneity settings, with larger gains observed when using a larger population size. These findings suggest that population-based server updates provide a promising alternative to standard single-model FL under non-IID data. Future work includes extending the evaluation to additional datasets such as CIFAR-100 and studying the behavior of FedEvo in broader federated settings.

ACKNOWLEDGMENT

This work was partially supported by the Institute of Industrial Technology (KEIT) grant funded by the Ministry of Trade, Industry, and Resources (MOTIR) under Grant RS-2025-02634277, Development of Human-AI Teaming-based Autonomous Security System for Future Cyber Threats, and by the Institute of Information & Communications Technology Planning & Evaluation (IITP) grant funded by the Ministry of Science and ICT (MSIT) under Grant RS-2026-25538638.

REFERENCES

[1] H. B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. Agüera y Arcas, "Communication-efficient learning of deep networks from decentralized data," in *Proceedings of the 20th International Conference on Artificial Intelligence and Statistics*, ser. Proceedings of Machine Learning Research, vol. 54. PMLR, 2017, pp. 1273–1282.

[2] P. Kairouz *et al.*, "Advances and open problems in federated learning," *Foundations and Trends in Machine Learning*, vol. 14, no. 1–2, pp. 1–210, 2021.

[3] T. Li, A. K. Sahu, M. Zaheer, M. Sanjabi, A. Talwalkar, and V. Smith, "Federated optimization in heterogeneous networks," in *Proceedings of Machine Learning and Systems*, vol. 2, 2020, pp. 429–450.

[4] S. P. Karimireddy, S. Kale, M. Mohri, S. J. Reddi, S. U. 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, vol. 119. PMLR, 2020, pp. 5132–5143.

[5] Y. Zhao, M. Li, L. Lai, N. Suda, D. Civin, and V. Chandra, "Federated learning with non-IID data," *arXiv preprint arXiv:1806.00582*, 2018.

[6] X. Li, K. Huang, W. Yang, S. Wang, and Z. Zhang, "On the convergence of FedAvg on non-IID data," in *International Conference on Learning Representations*, 2020.

[7] J. Wang, Q. Liu, H. Liang, G. Joshi, and H. V. Poor, "Tackling the objective inconsistency problem in heterogeneous federated optimization," in *Advances in Neural Information Processing Systems*, vol. 33, 2020.

[8] D. A. E. Acar, Y. Zhao, R. Matas, M. Mattina, P. Whatmough, and V. Saligrama, "Federated learning based on dynamic regularization," in *International Conference on Learning Representations*, 2021.

[9] A. Ghosh, J. Chung, D. Yin, and K. Ramchandran, "An efficient framework for clustered federated learning," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 19 586–19 597.

[10] F. Sattler, S. Wiedemann, K.-R. Müller, and W. Samek, "Clustered federated learning: Model-agnostic distributed multitask optimization under privacy constraints," *IEEE Transactions on Neural Networks and Learning Systems*, vol. 32, no. 8, pp. 3710–3722, 2021.

[11] A. Dieuleveut, G. Fort, E. Moulines, and G. Robin, "Federated expectation maximization with heterogeneity mitigation and variance reduction," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[12] O. Marfoq, G. Neglia, A. Bellet, L. Kameni, and R. Vidal, "Federated multi-task learning under a mixture of distributions," in *Advances in Neural Information Processing Systems*, vol. 34, 2021.

[13] T. Li, S. Hu, A. Beirami, and V. Smith, "Ditto: Fair and robust federated learning through personalization," in *Proceedings of the 38th International Conference on Machine Learning*, ser. Proceedings of Machine Learning Research, vol. 139. PMLR, 2021, pp. 6357–6368.

[14] M. Jaderberg, V. Dalibard, S. Osindero, W. M. Czarnecki, J. Donahue, A. Razavi, O. Vinyals, T. Green, I. Dunning, and K. Simonyan, "Population based training of neural networks," *arXiv preprint arXiv:1711.09846*, 2017.

[15] F. P. Such, V. Madhavan, E. Conti, J. Lehman, K. O. Stanley, and J. Clune, "Deep neuroevolution: Genetic algorithms are a competitive alternative for training deep neural networks for reinforcement learning," in *International Conference on Learning Representations*, 2018.

[16] M. M. Rahimi, H. I. Bhatti, Y. Park, H. Kousar, and J. Moon, "EvoFed: Leveraging evolutionary strategies for communication-efficient federated learning," in *Advances in Neural Information Processing Systems*, vol. 36, 2023.

[17] L. L. Custode, I. De Falco, A. Della Cioppa, G. Iacca, and U. Scafuri, "NEvoFed: A decentralized approach to federated neuroevolution of heterogeneous neural networks," in *Proceedings of the Genetic and Evolutionary Computation Conference*, 2024, pp. 295–303.

[18] L. L. Custode, G. Iacca, I. De Falco, U. Scafuri, and A. Della Cioppa, "Model-free-communication federated neuroevolution," *ACM Transactions on Evolutionary Learning and Optimization*, 2025.

[19] M. Hu, Z. Yue, X. Xie, C. Chen, Y. Huang, X. Wei, X. Lian, Y. Liu, and M. Chen, "Is aggregation the only choice? federated learning via layer-wise model recombination," in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, 2024, pp. 1096–1107.

[20] M. Hu, Y. Cao, A. Li, Z. Li, C. Liu, T. Li, M. Chen, and Y. Liu, "FedMut: Generalized federated learning via stochastic mutation," in *Proceedings of the AAAI Conference on Artificial Intelligence*, vol. 38, no. 11, 2024, pp. 12 528–12 537.

[21] A. Giuseppi, L. Della Torre, D. Menegatti, and A. Pietrabissa, "Adafed: Performance-based adaptive federated learning," in *Proceedings of the 5th International Conference on Advances in Artificial Intelligence*, 2022, pp. 38–43.